NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

THE ISLAMIC STATE BATTLE PLAN: PRESS RELEASE NATURAL LANGUAGE PROCESSING

by

James R. Friedlein

June 2016

Thesis Advisor:	Lyn R. Whitaker
Second Reader:	Craig A. Whiteside

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2016	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE THE ISLAMIC STATE BATTLE PLAN: PRESS RELEASE NATURAL LANGUAGE PROCESSING	**5. FUNDING NUMBERS**
6. AUTHOR(S) James R Friedlein	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The purpose of this study is to develop methods to accelerate and enhance the analysis of Islamic State Movement text documents. We analyze a unique database collected by Dr. Craig Whiteside, which is comprised of nearly 3,000 open-source translated press releases from 2003–2014. Using Natural Language Processing tools, the text data is aggregated into a corpus and processed based on document term structure and frequency. In order to reduce analyst workload, we validate Whiteside's manual analysis and construct cross-validated generalized linear models to automatically classify documents into one of seven types. A cascade classification model outperforms all other models with a mean cross-validated misclassification rate of 5.71 percent. Islamic State Movement operational summaries are classified as type "Celebrate." We develop a layered algorithm based on regular expressions and location searches to extract critical information from each attack event and display the details on a map using a web-based interactive **R Shiny** application. With the ability to automatically classify Islamic State Movement text documents and visually interact with the data contained within those classified as type "Celebrate," analysts and decision makers are able to process and understand large amounts of text data more quickly and effectively.

14. SUBJECT TERMS Islamic State Movement, Islamic State of Iraq, ISIS, Islamic State, Natural Language Processing, text mining, corpus, generalized linear model, cascade, **R Shiny**, leaflet, data visualization	**15. NUMBER OF PAGES** 83
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

THE ISLAMIC STATE BATTLE PLAN: PRESS RELEASE NATURAL LANGUAGE PROCESSING

James R. Friedlein
Major, United States Marine Corps
B.S., United States Naval Academy, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2016

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The purpose of this study is to develop methods to accelerate and enhance the analysis of Islamic State Movement text documents. We analyze a unique database collected by Dr. Craig Whiteside, which is comprised of nearly 3,000 open-source translated press releases from 2003–2014. Using Natural Language Processing tools, the text data is aggregated into a corpus and processed based on document term structure and frequency. In order to reduce analyst workload, we validate Whiteside's manual analysis and construct cross-validated generalized linear models to automatically classify documents into one of seven types. A cascade classification model outperforms all other models with a mean cross-validated misclassification rate of 5.71 percent. Islamic State Movement operational summaries are classified as type "Celebrate." We develop a layered algorithm based on regular expressions and location searches to extract critical information from each attack event and display the details on a map using a web-based interactive **R Shiny** application. With the ability to automatically classify Islamic State Movement text documents and visually interact with the data contained within those classified as type "Celebrate," analysts and decision makers are able to process and understand large amounts of text data more quickly and effectively.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CETIS	Center for Terrorism and Intelligence Studies
DTM	Document Term Matrix
GLM	Generalized Linear Model
GTD	Global Terrorism Database
HPC	High Performance Computing
IBC	Iraq Body Count
IDF	Inverse Document Frequency
IQR	Interquartile Range
ISI	Islamic State of Iraq
ISVG	Institute for the Study of Violent Groups
NATO	North Atlantic Treaty Organization
NLP	Natural Language Processing
PCorpus	Permanent Corpus
PDF	Portable Document Format
PGIS	Pinkerton Global Intelligence Services
SMART	System for the Mechanical Analysis and Retrieval of Text
START	Study of Terrorism and Responses to Terrorism
TDM	Term Document Matrix
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
tm	text mining (R package)
VCorpus	Volatile Corpus
WGS84	World Geodetic System 1984

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Analysts and researchers today are struggling to process the large amounts of information that flows across the myriad of global domains. In the world of text data, manual analytic methods cannot keep pace with the amount of text being published. In an effort to alleviate some of the burden of manual approaches, we apply Natural Language Processing (NLP) tools to a unique database of text documents collected by Whiteside (2014). His collection includes nearly 3,000 open-source press releases and captured documents authored by the Islamic State Movement's official media outlets and translated from Arabic to English. Compared to other terrorism databases, Whiteside's collection methodology limits the scope of the database and avoids coding conflicts and standardization issues common in larger terrorism databases. In this study, we develop two primary analytic tools that attempt to alleviate the burdens of manual analysis. With the ability to automatically classify documents and visualize large amounts of text data, analysts and decision makers will develop greater understanding of their data at a more rapid pace.

Whiteside's collection of 2,926 documents is stored as files, one per document, in Portable Document Format (PDF) and compressed Microsoft Word ".docx" formats. These documents are converted into text files and combined to form a corpus, an object that contains the text and metadata (such as the name of the original file) for all 2,926 documents. The corpus is used in two ways. First, it is used to classify documents into one of seven types, and then to visualize the locations of terrorist attacks described in those documents. Each of these tasks requires a different approach to processing the corpus.

For the classification task, the text of the corpus is preprocessed in a deliberate order so that attributes such as capitalization and the presence of whitespace, punctuation, and commonly used words do not adversely influence our analysis. "Terms," which roughly corresponds to words or sequences of words, are captured in a Document Term Matrix (DTM) (Feinerer and Hornik 2015). A DTM has one row per document and one column per term. Its entries are the number of times a term appears in a document. We

construct two DTMs; for the initial DTM, used to construct our “Weak Classifier,” a term is defined as a single word. For the second DTM, used to construct our “Strong Classifier,” we include all single words as terms, as well as two- and three-word sequences (often called bi-grams and tri-grams, respectively). To limit the size of the second DTM, the number of terms is reduced by removing those terms that appear in less than 1.0 percent of the documents. We further modify the DTM by replacing frequencies with a set of weights called Term Frequency-Inverse Document Frequency (TF-IDF) commonly used in information retrieval and NLP (Weiss, Indurkhya, and Zhang 2010). This set of weights is used to measure the importance of a term in the document. Terms that appear often in a document are weighted more, but terms that appear in most documents (e.g., “attack,” “kill”) and hence have little predictive power carry less weight.

In order to build supervised models that use document text to classify a document into one of seven types, we leverage Whiteside’s manual classification of each document type as our response variable. Of the seven types of documents over 85 percent are type “Celebrate.” After building multiple cross-validated logistic regression models using both types of DTMs, we determine that a cascade model provides the best performance (Friedman, Hastie, and Tibshirani 2010). The cascade model, similar to that used by Viola and Jones (2004), uses a sequence of our “Weak Classifiers” to classify documents as “Celebrate” or “Not Celebrate.” At each step, those documents classified as “Celebrate” are removed. Documents are classified as “Celebrate” if their estimated “Celebrate” probability ($\hat{p}$) is greater than a certain threshold. This threshold is set high to guard against removing “Not Celebrate” documents. After three applications of binomial classification, the distribution of document types becomes relatively uniform and a “Strong Classifier” is used in a final cross-validated multinomial classification model to classify documents into one of the seven types. This cascade model outperforms all other models with a mean misclassification rate of 5.71 percent. Figure 1 shows the production process for creating the cascade model.

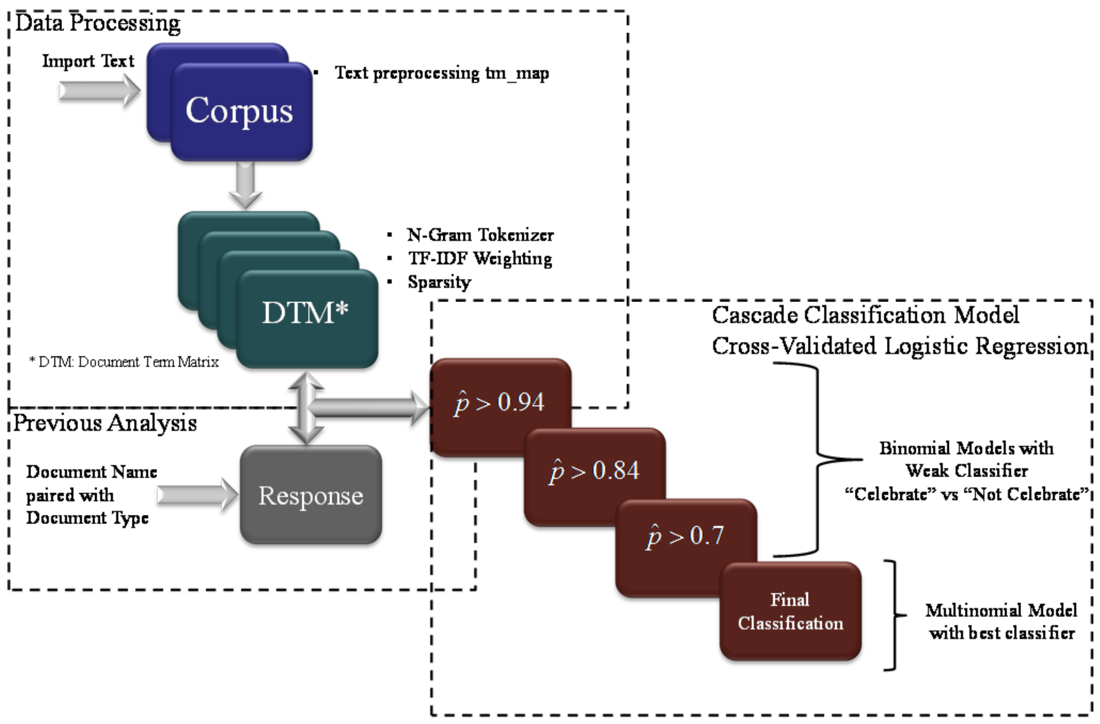
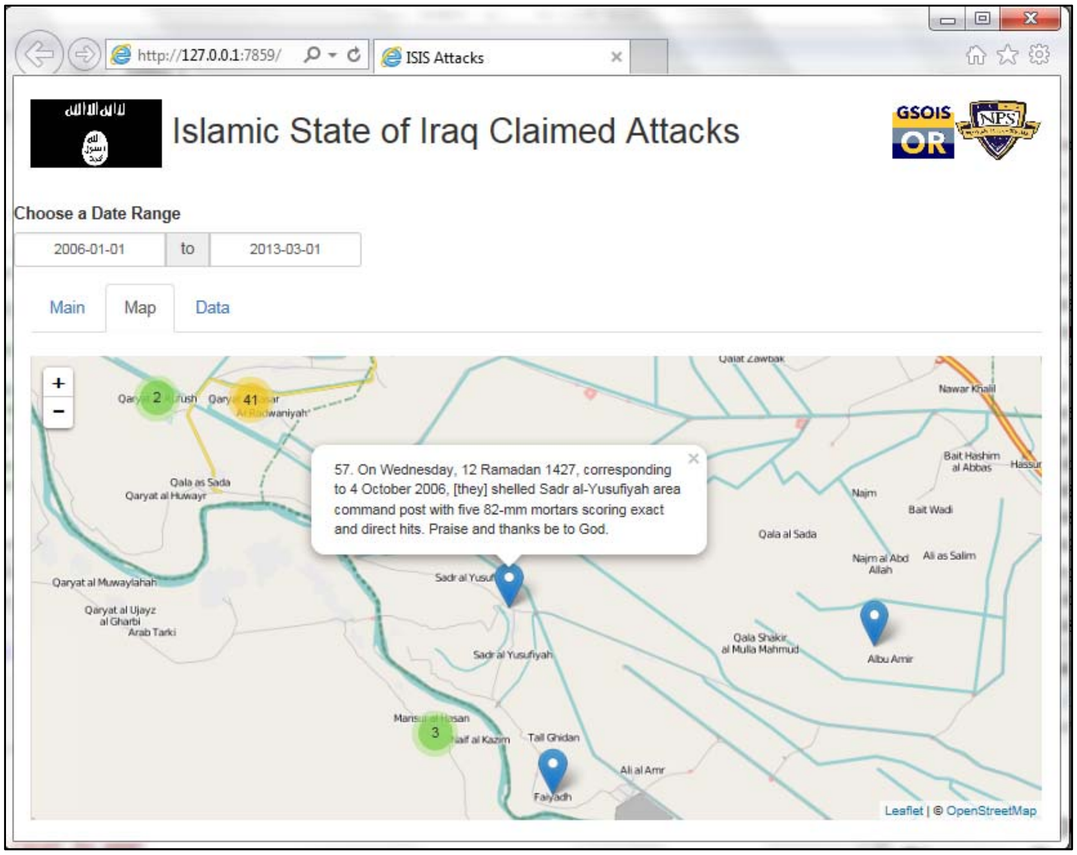

Data Processing
Import Text
Corpus
Text preprocessing tm_map
N-Gram Tokenizer
TF-IDF Weighting
Sparsity
DTM*
* DTM: Document Term Matrix
Previous Analysis
Document Name paired with Document Type
Response
Cascade Classification Model
Cross-Validated Logistic Regression
$\hat{p} > 0.94$
$\hat{p} > 0.84$
$\hat{p} > 0.7$
Final Classification
Binomial Models with Weak Classifier "Celebrate" vs "Not Celebrate"
Multinomial Model with best classifier

Figure 1: Diagram of Production Process for Cascade Classification Model

The documents classified as type "Celebrate" are Islamic State Movement operational summaries filled with information, to include dates, times, locations, and descriptive account of attacks. These documents provide an opportunity to develop an interactive tool to help analysts and decision makers visualize aspects of this rich source of information. Within the **R** interface, regular expressions are used to extract individual attacks or events from each "Celebrate" document and assign an event to its own row in a data set (R Core Team 2015). Functions leveraging regular expressions also extract and categorize event dates, times, and locations. Because locations are text spelled in different ways and can be names of streets, neighborhoods, areas, or other categories of location, location extraction is the most challenging task. We construct an algorithm that prioritizes five search functions based on the fidelity of the location category and whether the location is in a reference database that maps location names to geographical coordinates. The reference database is the GeoNames database, an open-source unclassified geographical database with approximately 32,000 locations (GeoNames 2016). In addition to the five location search categories, we augment the process by conducting a search for every primary and alternate name from the GeoNames database within all event text strings. We utilize the Naval Postgraduate School (NPS) Hamming supercomputer cluster to execute this 44-hour search in less than 2 hours and 45 minutes. We are able to provide location names for 91.2 percent of the 13,798 total events and coordinates for 42 percent of those events.

We then develop a web-based interactive map-overlay visualization tool by utilizing the **Shiny** and **leaflet** packages within **R** (Chang, Cheng, Allaire, Xie, and McPherson 2016, Cheng and Xie 2016). The tool's primary feature is the Map tab, which provides an interactive map with zoom functionality and location markers for each event. When the user clicks on a location marker, the original event text is displayed. The tool also provides functionality to select specific date ranges as well as display, search, and export from the source data frame. The **R Shiny** Map tab is shown in Figure 2.


http://127.0.0.1:7859/
ISIS Attacks
Islamic State of Iraq Claimed Attacks
GSOIS OR
NPS
Main
Map
Data
Leaflet | © OpenStreetMap

Choose a Date Range

2006-01-01 to 2013-03-01

57. On Wednesday, 12 Ramadan 1427, corresponding to 4 October 2006, [they] shelled Sadr al-Yusufiyah area command post with five 82-mm mortars scoring exact and direct hits. Praise and thanks be to God.

Figure 2: Map Tab for **R Shiny** Interactive Tool

With the ability to automatically classify Islamic State Movement text documents and visually interact with the data contained of those classified as type “Celebrate,” analysts and decision makers are able to process and understand large amounts of text data more quickly and effectively.

References

Chang W, Cheng J, Allaire J J, Xie Y & McPherson J (2016). Shiny: Web Application Framework for R. R package version 0.13.2. https://CRAN.R-project.org/package=shiny.

Cheng J, & Xie Y (2016). leaflet: Create Interactive Web Maps with the JavaScript “Leaflet” Library. R package version 1.0.1. https://CRAN.R-project.org/package=leaflet.

Feinerer I, Hornik K (2015) Text Mining Package “tm,” Version 0.6-2. (Jul 3) https://cran.r-project.org/web/packages/tm/tm.pdf.

GeoNames Geographical Database. Iraq (IQ) locations. Accessed May 1, 2016, http://download.geonames.org/export/dump/.

R Core Team (2015) R: A language and environment for statistical computing [Computer software]. Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org.

Viola P, Jones MJ (2004) Robust Real-Time Face Detection. *International Journal of Computer Vision.* 57(2): 137–154.

Weiss SM, Indurkhya N, Zhang T (2010) *Fundamentals of Predictive Text Mining* (Springer New York Inc., New York).

Whiteside C (2014) The Smiling Scented Men: The Political Worldview of the Islamic State of Iraq, 2003–2013 (December) Doctoral dissertation, Washington State University.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I could not have completed this thesis or the NPS Operations Research curriculum without the steadfast support and encouragement of my loving family. Jenny, thank you for allowing me to attend to my studies, even when it was inconvenient for you. We have faced a windy road called foster care during our time in Monterey, and despite significant demands on your time and emotions, you have always supported and encouraged me. I am also thankful to my boys, David and Aiden, who have been sensitive and understanding to the demands on my time, even when I was home.

I am thankful for the guidance of Professor Whitaker, who was willing to entertain my ideas and explore newer areas of research. Thank you for sharing your experience and your busy schedule with me. I am also thankful for Professor Whiteside and his Islamic State Movement expertise. Your experience and insights were helpful and your enthusiasm was encouraging as I set out on this journey with your dissertation and database in hand.

I also want to thank my fellow cohort members who helped me through my time at NPS. I am appreciative to Ben McCaleb for his help with numerous classes and projects, as well as Dave Macey for his assistance learning **R Shiny**. We have a great cohort and I was proud to share my time with such talented and selfless officers.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION AND BACKGROUND

A. INTRODUCTION

Analysts and researchers today are struggling to process the large amounts of information that flows across myriads of global domains. In the world of text data, manual analytic methods cannot keep pace with the numbers of documents and amount of text being published. In an effort to alleviate some of the burden of existing manual approaches, we apply Natural Language Processing (NLP) tools to a unique database constructed from approximately 3,000 English translated press releases published by the Islamic State Movement from 2003 to 2013.

This database was constructed manually by Whiteside (2014). It is unique in that it contains files with English translated text documents (one for each press release) and a Microsoft Excel Workbook containing data (by document), such as the date, author, document type, target, and type of incident extracted by a careful reading of each document. This database is also unique in that, unlike other terrorist databases, its scope is restricted to the Islamic State Movement terrorist activities, and only those activities as reported by the Islamic State's official media outlets. Thus, the perspective of the database is from one source, that of the perpetrators. These features of Whiteside's database make it an ideal vehicle for developing NLP tools to help automate the efforts of analysts to extract data from similar documents. In this thesis, we use Whiteside's database and NLP tools for two main purposes. The first purpose is to develop a classification model that demonstrates the ability to determine a document type based on its text contents. The second is to create an interactive web-based data visualization tool that assists analysts and decisions makers in their efforts to analyze terrorist events within time and geographic space.

In this chapter, we provide background by examining two widely used publically available databases of terrorist activities and comparing the methodologies used to construct these databases with the methodology used by Whiteside (2014) in constructing his Islamic State Movement database.

B. TERRORISM DATABASE METHODOLOGIES

Collecting data on terrorist activities is a challenging endeavor. Many obstacles present themselves when attempting to standardize data collection methodologies for loosely defined people and organizations committing violent acts for various goals. One of the first challenges is defining and understanding the types of activities that must be collected. The definition of terrorism is not always well understood and requires clarification. Should data collection include attempted acts of violence from non-state actors? Does there need to be a political incentive to an act of terrorism? Answering these and similar questions might not be of particular consequence in some cases, but it is important for the analyst and decision maker to understand the story behind the data when attempting to draw conclusions or make comparisons.

A second challenge is in the identification and verification of the data source. When collecting information on domestic criminal violence there are relatively few sources for collection. These include law enforcement "official" accounts, witness accounts, and self-reporting (LaFree and Dugan 2007). However, the majority of these types of sources do not exist for terrorist activities. It is essential that users accessing a terrorism database understand what sources are used in the data collection efforts and the validity of those sources. Multiple sources may report the same events and a mechanism must be in place to rectify conflicts or remove duplicates.

As we proceed with analysis in subsequent chapters, it is important that the scope, context, and collection methodology is thoroughly evaluated and understood. To this end, two current terrorism databases are presented and compared with Whiteside's Islamic State Movement database. These databases are the Global Terrorism Database (GTD) and the Iraq Body Count (IBC) Database.

1. Global Terrorism Database

The GTD is an open-source database that compiles information on global terrorist activities. It is administered by the National Consortium for the Study of Terrorism and Responses to Terrorism (START), which falls under the University of Maryland and the U.S. Department of Homeland Security (GTD 2016). GTD maintains a comprehensive database on terrorist events from 1970 through 2014. With a global outlook, the scope of the GTD is extremely large. It relies on a baseline database called the Pinkerton Global Intelligence Services (PGIS), which collected terrorist activities from 1970 to 1997. Critiques of the database point out that the number of incidents is very high. The GTD includes threatened attempts of violence in addition to violent acts themselves. In addition, LaFree et al. (2007), the director of START openly admits, "Because the goal of the data collection was to provide risk assessment to corporate customers, the data base was designed to err on the side of inclusiveness."

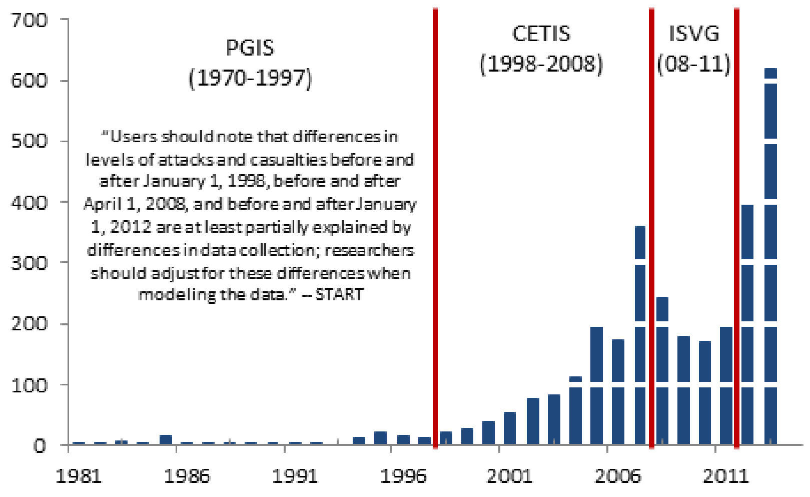
Starting in 1998, GTD began collecting its data in cooperation with the Center for Terrorism and Intelligence Studies (CETIS). However, the events were still captured retrospectively by identifying archived sources that document terrorist activities. In addition, GTD changed its definition of terrorism and expanded the number of variables it collected on each incident (GTD 2016). In 2008, the Institute for the Study of Violent Groups (ISVG) started a data collection process that recorded terrorist activities until October 2011. This collection was integrated with both the older and newer GTD collection methodologies (GTD 2016).

Since 2007, GTD has developed a new methodology for collecting virtually real-time terrorist activity data. As Jensen (2013), a GTD data collection manager explains,

> Instead of relying on the common practice of performing targeted searches against a few well-known news sources or news aggregators—a technique that was used in the past by vendors tasked with compiling the GTD—the GTD team ultimately decided that the best way forward was to start with an extensive pool of news articles culled from myriad sources.

This methodology has its pros and cons. Due to the vast amount of data being collected on a daily basis, GTD relies on tailored search terms to download "on average 1.3 million news articles per day from a pool of over 55,000 unique sources" (Jensen 2013). These sources are then processed with a series of NLP and machine learning tools in order to remove duplicates, find similarities between sources, and make an automatic determination of which sources contain terrorist events. These tools are likely effective in acquiring sources related to suspected terrorist events, but the fidelity and authenticity of the final classifications of the events have been called into question, especially compared with the methodology employed from 1997 to 2007.

A majority of GTD's critics point out that this comprehensive database is confusing and deceptive to the user because of its changing definitions and search methodologies, which impacts consistency in reporting. Pape, Ruby, Bauer, and Jenkins (2014) explain that "what START has not done is present a plan or even a commitment to resolve the underlying incongruity in collection methods across time that is painting a misleading picture of terrorism trends to the world." Although GTD's website attempts to maintain transparency by explaining its evolving processes and definitions, critics point out the impact these changes have had on numerical results. For example, Figure 1, taken from Pape et al. (2014), shows how the number of attacks recorded in GTD change over time and with the changes of collection methodology.



Number of terrorist attacks are shown by year and display changes in the GTD Data Collection Methodology from 1970 to 2014.

Figure 1. Number of Terrorist Attacks Recorded in the Global Terrorism Database. Source: Pape et al. (2014).

2. Iraq Body Count Database

The Iraq Body Count (IBC) database is an open-source web-based citizen initiative that describes itself as "rooted in the mainstream peace movement, which primarily opposes wars for being unnecessary" (Sloboda 2006). The scope is much different from the scope of the GTD, as it only focuses on Iraqi civilian deaths. IBC defends its data collection methodology that relies solely on open-source web-published media reports in the English language. It denies any bias introduced by limiting sources to English language media reports. IBC critics claim that its body counts significantly underestimate reality. A 2006 Lancet study estimates the number of deaths in Iraq from March 2003 to September 2004 to be 655,000 (Burnham, Lafta, Doocy, and Roberts 2006). In a September 2007 press release, IBC criticized the Lancet study stating that "our own view is that the current death toll could be around twice the numbers recorded by IBC and the various official sources in Iraq. We do not think it could possibly be 10 times higher." Critiques of the 2006 Lancet study cite issues with its methodology, to include its assumptions regarding the pre-invasion mortality rate and the number of sampling clusters used in the survey analysis (Kaplan 2006; Moore 2006). Spagat (2010) presents evidence that the study authors used data fabrication and falsification, and committed "ethical violations to the survey's respondents."

Despite its critics, IBC is considered a reputable reporting source for its area of expertise. It is believed by many to resist the tendency to over-report incidents and to maintain a consistent data collection methodology. IBC is often quoted in literature and used as a source for Iraqi civilian death counts and analysis.

3. Database Comparisons Illustrated

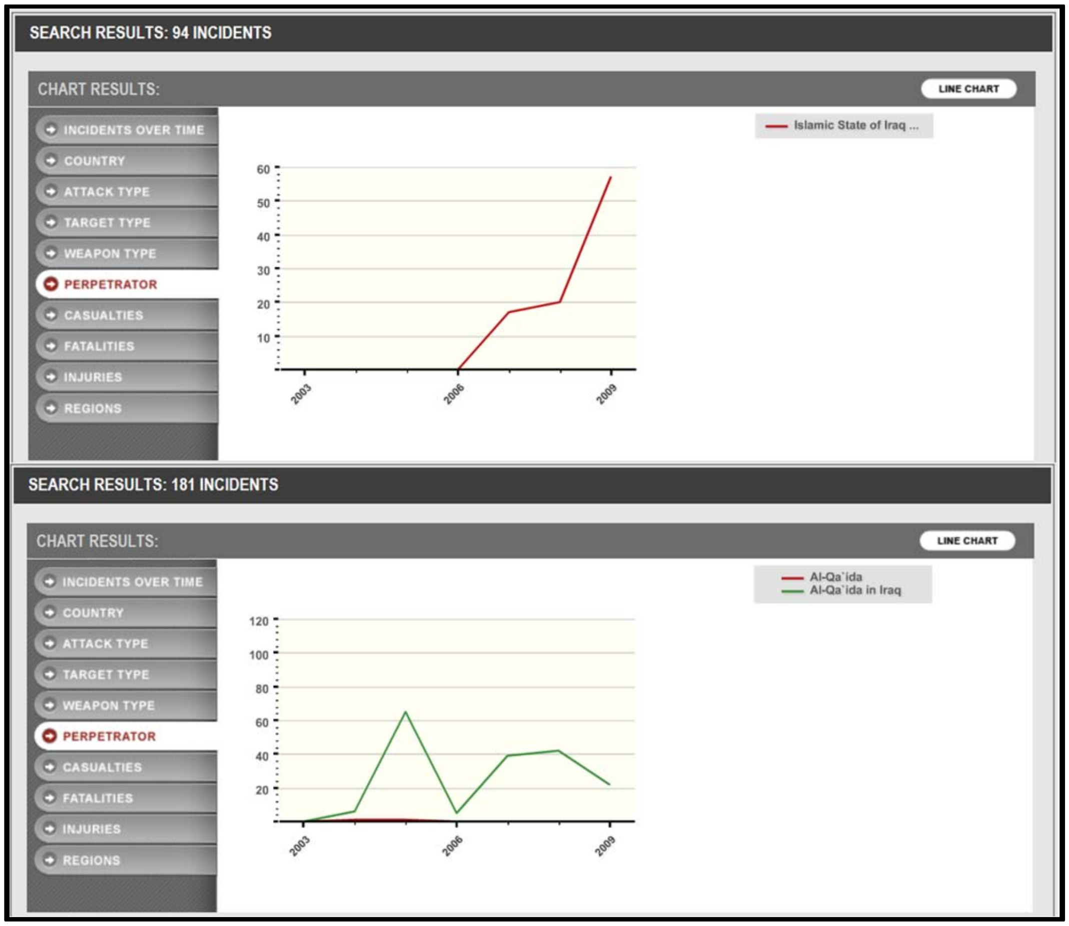
The reason for significant differences among reputable studies and databases often relies on their data collection and coding methodologies. IBC uses more restrictive search methods for its data collection and their definition criterion for an incident is slightly different from that of the GTD. The variables and coding methodology employed by IBC are also different from the GTD. IBC attempts to collect 18 variables on each incident whereas GTD collects between 45 and 120 variables. In addition, the IBC database can only be queried for one of six categories of perpetrators, of which specific terrorist groups cannot be selected. Within the GTD, the user has more flexibility to select a specific group to which the terrorist act can likely be attributed. IBC output focuses more on the statistics related to the number of incidents and number of individuals killed, rather than the group responsible for these actions.

The differences in terrorism databases are often not transparent to the user. For example, the methodology for coding the identity of a particular terrorist group can have significant consequences for an uninformed user. To illustrate this point we consider the coding methodology for the Islamic State Movement in the GTD from 2003 to 2009, but first it is important to understand the context of this example.

The Islamic State Movement can trace its origins to a group named Al-Qaeda in Iraq led by the Jordanian militant, Abu Musab al-Zarqawi, who sought refuge from Afghanistan in Iraq. During the spring of 2003, the United States invaded Iraq, and Zarqawi "collected a small group of true believers and awaited the coming invasion with a preternatural instinct for future opportunity" (Whiteside 2014). As the United States removed Saddam Hussein from power and relieved many Iraqi police and military from their duties, Zarqawi and his Sunni group seized the opportunity to gain power and influence. Although Zarqawi was killed in an airstrike in Iraq in 2006, the group's rise to prominence allowed them to gain acceptance as an Al-Qaeda affiliate and official recognition as the Islamic State of Iraq (Laub and Masters 2013).

The rise of the Islamic State Movement displays the complexities related to coding conventions for terrorist databases. The exact point in time when the Islamic State came to power and broke its affiliation from Al-Qaeda is not perfectly clear, but there is evidence that the Islamic State's identity formed under Zarqawi in 2003. It is more clear that the organization existed in October 2006, when the "Mujahidin Shura Council announced an alliance of several more jihadi factions and Sunni tribal leaders known as the Alliance of the Scented Ones" and subsequently announced the existence of the Islamic State of Iraq (Bunzel 2015). Despite this announcement, many journalists, reporters, and commentators continued to carry the name Al-Qaeda to represent the Islamic State for some time.

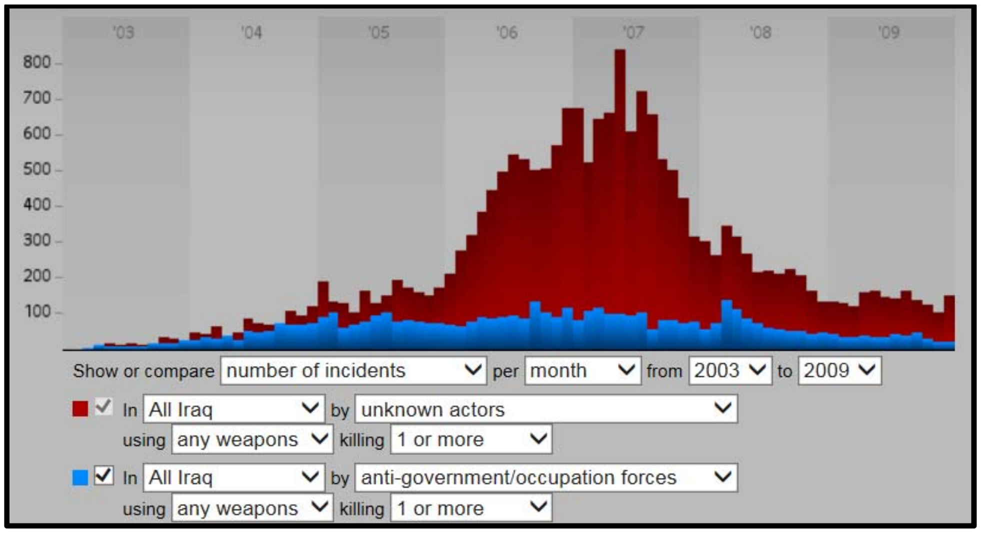
Two searches of the GTD illustrate the impact of naming conventions on database queries for the Islamic State Movement from 2003 to 2009. A first search yields 94 incidents conducted by the "Islamic State of Iraq," of which zero incidents are attributed to the group prior to 2006. A second search shows 181 total incidents conducted by the groups "Al-Qa'ida" and "Al-Qa'ida in Iraq." Of these 181 incidents, 103 occur from 2007 to 2009. The GTD coding shows that the database fails to clearly delineate the transition of the terrorist group from its status as Al-Qaeda to the Islamic State of Iraq. A final inclusive search for all perpetrator categories from 2003 to 2009 reveals 5,144 incidents, of which over 90 percent belong to the "Unknown" perpetrator group. Figure 2 shows screen shots of the results of the first two searches.



All GTD Searches: Iraq, 2003–2009
Top Search: "Islamic State of Iraq": 94 Incidents
Bottom Search: "Al-Qa'ida" and :Al-Qa'ida in Iraq": 181 Incidents
Additional Search (not depicted): "Unknown" perpetrator: 5,144 Incidents

Figure 2. Global Terrorism Database Search Results for Al-Qaeda and the Islamic State of Iraq from 2003–2009. Source: GTD (2016).

A similar search of the IBC database from 2003 to 2009, shown in Figure 3, results in 5,002 incidents conducted by "anti-government/occupation forces." This coding category for the incident perpetrator is most aligned with incidents attributable to the Islamic State Movement. Similar to the GTD, many incidents are not assigned to a specific perpetrator group.



Show or compare number of incidents per month from 2003 to 2009

🟥 ✓ In All Iraq by unknown actors
using any weapons killing 1 or more

🟦 ✓ In All Iraq by anti-government/occupation forces
using any weapons killing 1 or more

Blue: Incidents conducted by "anti-government/occupation forces": 5,002
Red: Incidents conducted by "unknown actors": 19,821

Figure 3. Iraq Body Count Database Search Results for Perpetrator Groups Likely to Contain Islamic State of Iraq Members, 2003–2009 Source: IBC (2016).

Disparities between databases concerning coding methodologies like categorization of perpetrator groups highlight areas of caution. If a database has a large scope and research is being conducted for a specific subset or focus area, it is important to first understand the database definitions, as well as data collection and coding methodologies. In this thesis, we will analyze a database very similar to that of the GTD and IBC, but of narrower scope and designed to represent a particular perspective.

4. Islamic State Movement Database

In December 2014, Dr. Craig Whiteside, a retired United States Army Lieutenant Colonel, published his Ph.D. dissertation titled, "The Smiling, Scented Men: The Political Worldview of the Islamic State of Iraq, 2003–2013." His research focuses on investigating the political worldview of this powerful terrorist organization in order to describe their strategic targeting (Whiteside 2014). Whiteside's research involves the analysis of nearly 3,000 written messages from the group's officially sanctioned media outlets. These messages written by the organization's most trusted agents attempt to capture and disseminate a focused and comprehensive account of the group’s strategic messaging and operational successes. A majority of the collected documents were translated from Arabic to English by the United States government and made available on an open-source website. Whiteside (2014) augments the collection from the following sources and collections: Combating Terrorism Center (CTC) Harmony, Haverford College, Flashpoint, and the GTD.

Whiteside (2014) developed his database over time through careful manual analysis of document sources. Documents written from the Iraqi Government or North Atlantic Treaty Organization (NATO) perspective, for example, were discarded in the interest of developing a particular data collection methodology. His desire was to capture the true Islamic State Movement perspective by using all messages from their officially sanctioned media sources. Whiteside explains, “Overall, the data I have collected is reasonably comprehensive, with an estimated 90% of all press releases ISI [Islamic State of Iraq] has distributed.”

The scope of Whiteside’s Islamic State Movement database is limited to a single perpetrator, which provides an advantage over other large databases. This allows the sources to be vetted for authenticity in the collection process. As seen in other databases, attributing attacks to the Islamic State can be very difficult, especially as many incidents do not assign a perpetrator or assign them under different names. This is particularly critical for the 2003–2009 time period in Iraq, where there is disagreement over the appropriate name for Zarqawi’s emerging organization. Whiteside’s database attributes activities in 2003 to the Islamic State Movement and consistency is maintained throughout the entire database from 2003 to 2014.

Because the point of view represented in Whiteside’s database is limited to sanctioned Islamic State Movement media sources such as Al-Fajr Media, critics may point out that the database contains exaggerated accounts of terrorist activities. The idea that a terrorist group inflates its victories in order to spur on its members is a real concern. However, the unique nature of Whiteside’s collection is its resolve to preserve the Islamic State Movement point of view to better understand the political worldview and motivations of this highly influential terrorist organization. Analysis from this collection can also be used to tell a story rarely told in the western world. Through analysis, the Islamic State Movement narrative can be replayed and compared to other accounts, and even challenge accepted historical records or point to areas of history that may need to be revisited.

C. OUTLINE

Subsequent chapters provide details on how to apply NLP tools to the Islamic State Movement database. Chapter II describes methods to import text from our database documents and preprocess it in preparation for model building. Chapter III details the development and results from multiple cross-validated classification models that automatically classify a document’s type. Chapter IV takes the documents classified as type “Celebrate,” extracts event summaries, and constructs a data visualization tool that provides a web-based interactive map containing the plotted events. Finally, Chapter V provides a summary and recommendations for future work.

II. DATA PROCESSING

A. ISLAMIC STATE MOVEMENT DATABASE

The database collected by Whiteside is archived as a set of files (one per document) and organized into multiple directories by year and type of document. Ninety-nine percent of the 2,926 documents are Islamic State press releases from open-source official media outlets. These press releases were published for a variety of purposes and represent the official messaging of the terrorist group. The database includes 28 translated captured documents that detail what the Islamic State leadership is trying to convey to its members. It also includes 41 leadership statements from the Haverford Collection and 15 documents from other sources. Of particular interest for subsequent processing of document text are the individual document formats and any inherent encryption or file compression. The two document formats, Portable Document Format (PDF) and Microsoft Word docx, will be discussed in more detail in the next section. Table 1 details the breakdown of the database by directory organization and file format.

Table 1. Islamic State Database Breakdown by Document Format

Source	**Open Source Press and Media Releases**										**Other Collections**		
Document Format	**PDF**		**docx**								**PDF and docx**		
Directories	**2004**	**2005**	**2006**	**2007**	**2008**	**2009**	**2010**	**2011**	**2012**	**2013**	**Captured Documents**	**Haverford Leadership Statements**	**Misc**
No. Documents	80	143	1115	891	191	115	102	88	92	25	28	41	15
Total Documents	**2926**												

B. DATA IMPORT

The amount of data contained in a set of 2,926 documents of varying lengths may be overwhelming to certain data structures and hardware systems. One of the main data tools for managing large amounts of text from multiple documents is called a corpus. The **R** software text mining package, **tm**, is tailored specifically to create and analyze corpora and other text mining structures that have been developed for NLP (R Core Team 2015; Feinerer and Hornik 2015). The default corpus type for the **tm** package is the Volatile

Corpus (VCorpus), which can be stored as an **R** object on a local machine. Larger corpora can be imported as Permanent Corpus (PCorpus), which are stored outside of **R** on an external database (Feinerer and Hornik 2015).

The **tm** package has the functionality to read documents of various formats into a corpus. As shown in Table 1, the Islamic State Movement database has PDF files and zipped compressed Microsoft Word documents with extension docx. The `readerControl` argument of the **tm** function `VCorpus()`allows the user to specify file formats of the documents to be imported as a `VCorpus` object. The option `readPDF` for the argument `readerControl` is used to import the PDF files. However, the option `readDOC` uses a Microsoft Word reader that relies on *antiword* software that can only convert documents from Microsoft Word versions 2003 or earlier to plain text (van Os 2008). It cannot be used on Microsoft docx files, which store document information using the Open XML Format and require the functionality to unzip compressed files and folders.

In order to process docx files, it is necessary to use software capable of unzipping and converting these files to an alternate format. A free open-source Perl-based utility named docx2txt is used to convert docx files to text files (Kumar 2014). Docx2txt is a command-line utility that includes a Perl script, a Unix/Windows wrapper script and a configuration file. A sample **R** script using the doxc2txt to convert docx document files to text files is shown in Appendix A.

In order to create a combined VCorpus object that contains all 2,926 documents several methods are considered. One option is to import all PDF documents into a corpus and then import all text files into a separate corpus. The two corpora can then be combined into one. However, we chose to convert all documents (both PDF and docx files) to text files and then import all text files into one corpus utilizing the `readPlain` option for the `readerControl` argument.

To convert the PDF documents into text files, we again leverage a free open-source software product named Xpdf (Glyph and Cog 2014). The Xpdf suite includes a PDF text extractor. It is essential to ensure that the PDFs to be converted do not contain password security or encryption, as the Xpdf package will honor these permission settings. A simple **R** script similar to the docx2txt script is utilized to convert all PDF's to txt files. A sample **R** script is given in Appendix A.

With all documents appropriately formatted as text documents, and stored as files with .txt extensions in multiple directories, a VCorpus is constructed by passing to the `VCorpus()` function a character vector with the names of all directories containing the 2,926 text files. The resulting corpus contains the text from all documents as well as metadata. The metadata includes the date-time stamp, derived from when the document was uploaded to the corpus, as well as the text language and id tag, which contains the document file name.

C. TEXT PREPROCESSING

Understanding the type of text contained within a corpus is critical to establishing a logical and ordered path for text processing. Our analysis relies on a "bag of words" approach in which each document is considered to be an unordered collection of words or tokens. Thus, the text preprocessing step involves making decisions about how to tokenize or identify groups of letters as tokens, which tokens should be ignored, and how to deal with numbers, punctuation and other attributes of the text.

Text attributes of whitespaces and the use of capital letters, numbers, and stopwords are often considered in preprocessing of the text. We remove extra whitespace from the corpus, as extra spaces serve no benefit in identifying tokens. Capital letters can be used to identify proper names, but inconsistencies with text capitalization make this difficult. Another common practice that we follow is to eliminate capital letters by changing all text to lower case. Numbers may also be useful in certain contexts but they often provide undesirable clutter. We also remove numbers in the text preprocessing step. Stopwords, such as "a," "an," and "the," are commonly used words in English that provide little added benefit to the understanding of the text. There are variations of stopwords for each language. The System for the Mechanical Analysis and Retrieval of Text (SMART), an information retrieval system developed by Cornell in the 1960s, developed a list of English stopwords, which we utilize (SMART information retrieval system 2004).

Another common text processing technique is to "stem" words. Stemming reduces words to their root form. For example, in our corpus, it is desirable to stem the words "assassinate," "assassinated," and "assassins" to the word "assassin" so all variations of this word are treated as a single term. **R**'s **tm** package utilizes Porter's Stemming Algorithm (Porter 1980) for term normalization in information retrieval systems.

Text processing is an iterative process, which involves conducting a sequence of transformations on the corpus and evaluating its subsequent characteristics. Inspecting the resultant types of tokens and their frequency is critical to iterative evaluations. We highlighted the role of punctuation in our corpus for future analysis. It becomes evident that most punctuation is undesirable. However, many Arabic names and locations rely heavily on apostrophes and dashes. Therefore, we construct a function `RemoveMostPunctuation()`, found in Appendix C, that removes all punctuation except for apostrophes and intra-word dashes.

This specialized function in combination with the `tm_map()` function from the **tm** package is used in a specific sequence of steps to preprocess the text. To illustrate the corpus preprocessing steps, a line of text is selected from the corpus and the results of each step of the process are displayed in sequence.

Text from original corpus
On Wednesday, 9 Muharram 1427, corresponding to 8 February 2006, your brothers from the military wing of The Mujahidin Shura Council assassinated one of the members of the apostate police named Ali Hariqah in Al-Tahrir area in Ba'qubah.

Change to lowercase and remove numbers (using `tm_map()`)
on wednesday, muharram , corresponding to february , your brothers from the military wing of the mujahidin shura council assassinated one of the members of the apostate police named ali hariqah in al-tahrir area in ba'qubah.

Remove "SMART" stopwords (using `tm_map()`)
wednesday, muharram , february , brothers military wing mujahidin shura council assassinated members apostate police named ali hariqah al-tahrir area ba'qubah.

Remove most punctuation (using `RemoveMostPunctuation()`)
wednesday muharram february brothers military wing mujahidin shura council assassinated members apostate police named ali hariqah al-tahrir area ba'qubah

Stem and strip whitespace (using `tm_map()`)
wednesday muharram februari brother militari wing mujahidin shura council assassin member apost polic name ali hariqah al-tahrir area ba'qubah

Another example illustrates the benefit of preserving dashes and apostrophes while removing most punctuation from the corpus. A selection of 16 terms from the corpus processed using the `RemoveMostPunctuation()` function shows how many Arabic names and locations are preserved.

"al'uyasat"	"kurdiyah"	"invador"	"al-tal"
"abdil-muhsin"	"factori"	"cavalri"	"mudaf"
"goal"	"al-biyah"	"disc"	"al'zamiyah"
"al-jinabi"	"al-khamisat"	"al-ruba'iyyah"	"al-mismari"

There is no perfect strategy for text preprocessing, but there are methods that work more effectively than other methods. Most preprocessing strategies come from a familiarity with the text and developing a sense for what terms prove the most useful in subsequent analysis. In the case of the Islamic State Movement corpus, we feel the ordered processing sequence described in this example, and coded in Appendix D, provide the most advantageous approach to preprocessing the text.

D. DOCUMENT TERM MATRIX

Once the text of each document of the corpus is preprocessed, the frequencies of each term by document are stored in a Document Term Matrix (DTM) or Term Document Matrix (TDM). A term can be a single token or a sequence of tokens. We construct two DTMs; both are constructed from the preprocessed corpus and contain 2,926 rows, with each row corresponding to a document. The initial DTM has 24,111 columns, corresponding to the frequencies of terms found in the corpus, where terms are defined as single tokens or uni-grams. In this section, we discuss modifications of this initial DTM.

1. N-Gram Tokenizer

Because the DTM of single tokens isolate tokens from their context, n-grams are a method to preserve some potentially useful relationships between adjacent tokens. N-grams are sequences of n consecutive tokens found in the documents of a corpus. The **RWeka** package provides an `NGramTokenizer()` function that enables the user to preserve groups of consecutive tokens based on a minimum and maximum phrase length (Hornik, Bucht, and Zeileis 2009). For the Islamic State corpus we construct a new DTM using the `NGramTokenizer()` function. The new DTM includes uni-grams, bi-grams, and tri-grams. This increases the size of the DTM from 24,111 terms to over one million terms.

2. Term Frequency-Inverse Document Frequency Weighting

The weightings applied to terms in each document prove to be influential in subsequent analysis. The Term Frequency (TF), x_{ij}, measures how often term i appears in document j. Because every document varies in length, the term frequency is normalized often by dividing by the number of terms in the document giving the document normalized TF (Equation 1) (Weiss, Indurkhya, and Zhang 2010):

$$\frac{x_{ij}}{\sum_j x_{ij}} . \tag{1}$$

However, often the terms that are most important are those that only appear in a few documents, whereas terms that appear in almost all documents are of little help in identifying unique characteristics of a document. The Inverse Document Frequency (IDF) is a decreasing function of the number documents in which a term appears (Weiss et al. 2010). IDF is defined for each term *i* and the definition we use for IDF_i (Equation 2), the *i*th term's IDF, is that used by **tm** (Feinerer and Hornik 2015).

$$IDF_i = \log_2 \left(\frac{D}{\sum_{j=1}^{D} I\left(x_{ij} > 0\right)} \right), \tag{2}$$

where *D* is the number of documents and *I* , the indicator function is 1 is its argument is true and 0 otherwise. The Term Frequency-Inverse Document Frequency (TF-IDF) weights combine (1) and (2) to give terms more weight that appear more frequently in a document, but less weight if they appear in most documents. $TFIDF_{ij}$ (Equation 3) is defined for term *i* and document *j* as

$$TFIDF_{ij} = \frac{x_{ij}}{\sum_j x_{ij}} IDF_i \quad . \tag{3}$$

The package **tm** provides a `weightTfIdf()` function with an option to compute $TFIDF_{ij}$, as in Equation 3. See *Fundamentals of Predictive Text Mining* (Weiss et al. 2010) for a more thorough discussion and variations of TF-IDF weightings.

3. Sparsity

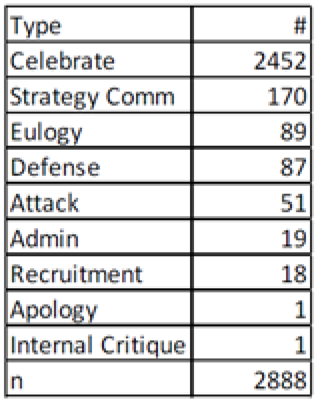
Another vital attribute of the DTM is its size. The initial DTM, which defines terms as uni-grams, is a manageable size of 9.7 MB. Introducing bi-grams and tri-grams as terms in the DTM increases the number of terms to over a million and the size of the matrix grows to over 113 MB. Two approaches to reducing the size of the DTM are explored. The first relies on functionality within the **tm** package to remove terms that occur in very few documents. The second approach is to store the DTM in a compressed manner by only storing non-zero entries and their locations.

One way to reduce the size of the DTM is to leverage the `removeSparseTerms()` function within the **tm** package. When applied to this function, the `sparse` argument refers to the threshold for the proportion of documents which do not contain a term, above which the term will be removed. We apply a `sparse` argument of 0.99 to our corpus, which means that only terms which appear in fewer than one percent of the documents will be removed. By changing the sparsity of the DTM to 99 percent, the number of terms in the matrix reduces from 1,007,655 to 40,926. This one percent change makes the matrix less sparse and provides a large decrease in matrix size, reducing the DTM from to 113MB to 36.5 MB.

R's **Matrix** package is used to compress the size of the DTM even further (Bates and Maechler 2015). The function `sparseMatrix()` represents the matrix by two vectors which identify the row and column indices of the non-zero terms within the matrix, as well as the values of the non-zero entries and the overall dimensions of the DTM. As a result, the data is further reduced from the 36.5 MB DTM to 16.9MB in sparse matrix format.

E. SUPERVISED MODEL PREPARATION

As Whiteside (2014) illustrates in his dissertation, the Islamic State Movement displayed various methods of communication in the database documents. A breakdown of his classification of documents is displayed in Figure 4. An overwhelming majority of the documents, the "Celebrate" documents, contain a message of celebration. These documents are mostly comprised of detailed operational summaries of successful attacks on military targets. “Strategic Comm(unication)” is another highly represented category. These documents are taken from video, audio, and written sources. They are often lengthy and written by top Islamic State leadership in order to disseminate strategic messages. Whiteside describes that the “Eulogy” documents are similar to “Recruitment” documents but maintain a heavy focus on memorializing martyrs. “Defense” documents are used to provide a rebuttal to public criticism, often related to civilian casualties. The “Attack” category describes documents in which the Islamic State identifies and criticizes specific individuals. Administrative documents, labeled “Admin,” and “Recruitment” documents are slightly less prevalent in the database. The “Apology” and “Internal Critique” documents are limited to one document each (Whiteside 2014).



Type	#
Celebrate	2452
Strategy Comm	170
Eulogy	89
Defense	87
Attack	51
Admin	19
Recruitment	18
Apology	1
Internal Critique	1
n	2888

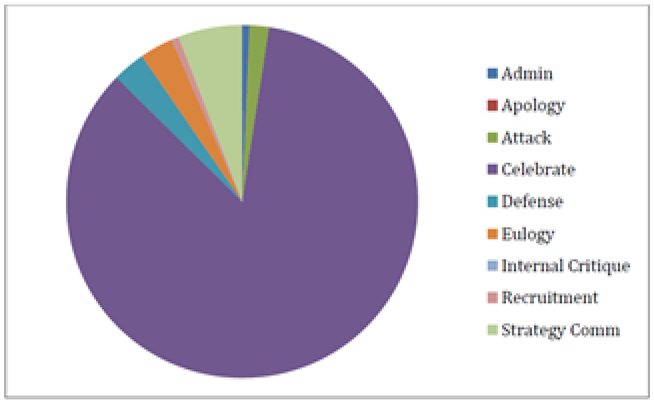


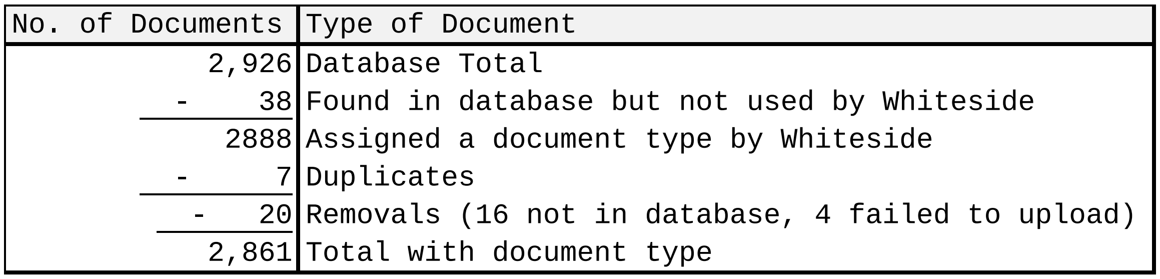
Figure 4. Whiteside’s Breakdown of Islamic State Movement Database by Message Type. Source: Whiteside (2014)

1. The Goal

The evidence of similarities between documents is an important concept captured by Whiteside’s research. Based on his expertise, each of the 2,888 documents in the database was manually evaluated and classified as one of the types of documents in Figure 4. An important question is whether NLP tools can identify the same characteristics as human expertise. In a broader application, the ability to classify massive amounts of documentation in an automated fashion based on a model will prove highly advantageous. Time-starved intelligence analysts searching for particular topics or a specific type of document could benefit tremendously from automation. In the next chapter, we develop a model to help automate Whiteside's manual classification.

2. Response Variable Reconciliation

In his research, Whiteside stored records of his analysis in Excel spreadsheets. These provide insights regarding his coding methodology for classifying documents in his database, as well as insights into the data contained within each document. As with any database, data entry errors and miscellaneous peculiarities must be reconciled. As noted in Table 1, the corpus built from the database contains 2,926 documents. Figure 4 reveals that Whiteside's analysis contains 2,888 documents. The breakdown of documents is displayed in Figure 5. Through data reconciliation, we discover that 38 documents were added to the database after Whiteside completed his analysis. By developing **R** code to compare document names between Whiteside's coded spreadsheet and the database of documents, we also find that seven coded entries are duplicates and 20 are missing and need to be removed. Of note, nine documents are preserved despite discrepancies in document names, such as punctuation and spelling. In the end, there are 2,861 document texts, each with a corresponding document type.



No. of Documents	Type of Document
2,926	Database Total
- 38	Found in database but not used by Whiteside
2888	Assigned a document type by Whiteside
- 7	Duplicates
- 20	Removals (16 not in database, 4 failed to upload)
2,861	Total with document type

Figure 5. Islamic State Data Set Document Breakdown based on Reconciliation with Whiteside's Coded Research

Table 2 gives the breakdown of the 2,861 document types. The "Warning" classification was added by Whiteside in later research due to its unique characteristics (Whiteside 2016). For our analysis, we remove three documents corresponding to the classifications "Warning," "Apology," and "Internal Critique." This leaves 2,858 documents for analysis. Of these, we note that over 85 percent are of class “Celebrate,” a feature that we address in the Chapter III.

Table 2. Islamic State Document Type Classifications as Coded by Whiteside

Admin	**Attack**	**Defense**	**Eulogy**
14	50	85	86
Recruitment	**Warning**	**Apology**	**Celebrate**
18	1	1	2447
Internal Critique	**Strategic Comm**		
1	158		

THIS PAGE INTENTIONALLY LEFT BLANK

III. CLASSIFICATION MODEL

A. MODELING APPROACH

In this chapter, we use the DTMs constructed in Chapter II to fit models to classify by document type. We begin the chapter by describing the approach of Friedman, Hastie, and Tibshirani (2010) for fitting generalized linear models, and in particular for fitting cross-validated multinomial logistic regression models for classification. We use their approach to fit two models. Model 1 is based on a uni-gram DTM. Model 2 is based on a DTM with added bi-grams and tri-grams and TF-IDF weights. Because the "Celebrate" documents account for 85 percent of the data, and hence drive both model fits, we introduce a cascade of models based on Model 1 and Model 2 that outperforms either model.

B. CROSS-VALIDATED GLMNET

Glmnet is an **R** package which fits a regularized generalized linear model by minimizing a penalized negative log-likelihood. The penalties include the lasso and ridge penalties, as well as a mixture of the two, called the elastic net. Lasso uses the L_1 norm of the coefficient vector as a penalty, whereas ridge uses the L_2 norm. One very helpful feature of **glmnet** is that it handles sparse features efficiently. The **glmnet** package also accommodates many families of response types, to include binomial, Gaussian, Poisson, and multinomial, as well as others. In our case, it is appropriate to use the multinomial response to classify the document types into one of seven categories (Friedman et al. 2010).

One effective method to estimate and reduce prediction error from the model is to use cross-validation. Cross-validation allows us to strike a balance between a model that is overfit with low bias and high variance and a model which is underfit with high bias and low variance (Faraway 2006). Although the cross-validated prediction error is biased, it helps identify a model with the appropriate tradeoffs. A function called `cv.glmnet()` in the **glmnet** package is used to conduct K-fold cross-validation (Friedman et al. 2010). Because the database is relatively small (2,861 documents), cross-validation enables us to use the whole database and alleviates the requirement of choosing and setting aside a validation set. In the case of our K=10 randomly-selected folds, `cv.glmnet()` fits the model to $K-1=9$ folds and predicts on the remaining fold. This process yields one prediction error, named cv_1. This process is repeated *K-1* times yielding the corresponding *K-1* prediction errors, $cv_2,...cv_K$. The average of the $K=10$ prediction errors produces the cross-validated error, cv (Equation 4) (Hastie, Tibshirani, and Friedman 2009):

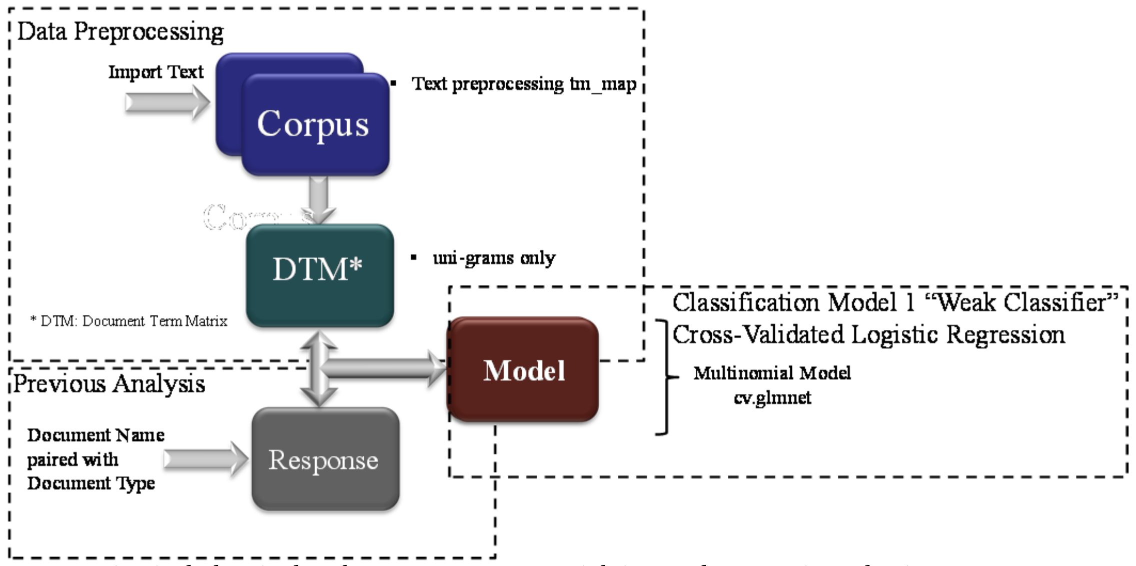
$$cv = \frac{\sum_{k=1}^{K} cv_k}{K} \quad . \tag{4}$$

We use the “one-standard error” rule to choose the best model from the varying subset sizes. This rule is often used with cross-validation, “in which we choose the most parsimonious model whose error is no more than one standard error above the error of the best model” (Hastie 2009).

Building classification models with **glmnet** is an iterative process which is highly influenced by the preprocessing detailed in Chapter II. Not only do preprocessing actions like term weighting and tokenization have an effect on the predictive power of the classification model, but the order of steps has an impact as well. By varying these procedures applied to the corpus and DTM, we are able to build a cross-validated generalized linear multinomial classification model and measure its performance. Performance is measured by the mean cross-validated misclassification rate for each model. For `cv.glmnet()` we note that the results of each model vary because the data are partitioned randomly in to $K=10$ folds. Thus, we present an example of our cross-validated model results and then provide a mean cross-validated misclassification rate based on one hundred runs of each cross-validated model (Friedman et al. 2010). Two refined model results are compared to illustrate the effects of data processing on performance.

1. Model 1: Weak Classifier

Data preprocessing of the corpus for Model 1 is conducted as shown in Appendix D. We change all letters to lower case, remove numbers and stop words, and preserve intra-word dashes and apostrophes. In addition, we stem all tokens and remove extra whitespace. Upon completion, we transform the corpus into a DTM. All terms are uni-grams, whose frequencies will serve as the independent variables for our multinomial cross-validated model. Because the size of the DTM without n-grams is manageable (9.7 MB), there is no requirement to make adjustments for sparsity. We use this DTM and the corresponding response variables to fit the first model. The process to generate the multinomial classification model based on single-token terms is depicted in Figure 6.


Data Preprocessing
Import Text
Corpus
▪ Text preprocessing tm_map
DTM*
▪ uni-grams only
* DTM: Document Term Matrix
Classification Model 1 "Weak Classifier"
Cross-Validated Logistic Regression
Multinomial Model
cv.glmnet
Model
Previous Analysis
Document Name paired with Document Type
Response

Preprocessing includes single-token DTM, no term weighting, and no sparsity reduction

Figure 6. Diagram of Production Process for Classification Model 1.

The results of one run of Classification Model 1 are displayed in Table 3. The cross-validated misclassification rate for this run is 8.96 percent. We see that the 2,447 "Celebrate" documents influence other document types, forcing them to also be classified as "Celebrate." For example, only two of the 50 "Attack" documents are properly classified and 38 of the 50 are misclassified as "Celebrate" documents. This is an issue that we will address in subsequent models.

Table 3. Model 1 Classification Table–Actual Versus Predicted

	Predicted Classification						
Coded Classification	Admin	Attack	Celebrate	Defense	Eulogy	Recruitment	Strategic Comm
Admin	3	0	11	0	0	0	0
Attack	0	2	38	2	0	0	8
Celebrate	0	0	2442	0	0	0	5
Defense	1	0	39	36	1	0	8
Eulogy	0	0	32	1	50	0	3
Recruitment	0	0	12	0	1	1	4
Strategic Communication	0	0	69	2	2	0	85

Cross-Validated Misclassification Rate: 8.96 percent.

The mean cross-validated misclassification rate over 100 runs is **8.937** percent. In subsequent models, it becomes clear that Model 1 is the weaker classifier.

To gain more insight into comparing similarities between documents based on Model 1, we observe the ability of the model to accurately classify “Celebrate” documents into the proper class. Figure 7 gives the distributions of the estimated probability that a document is of class “Celebrate” by document type. The estimated probabilities for the “Celebrate” documents tend to be high. In addition, there is a median of 0.60 for the estimated probability for “Admin” documents. In contrast, documents coded as types “Eulogy” and “Strategic Communication” have median estimated probabilities of less than 0.30. Based on Model 1, we are persuaded that “Celebrate” documents share more similar features with “Admin” and “Recruitment” documents than those coded as “Eulogy” or “Strategic Communication.”

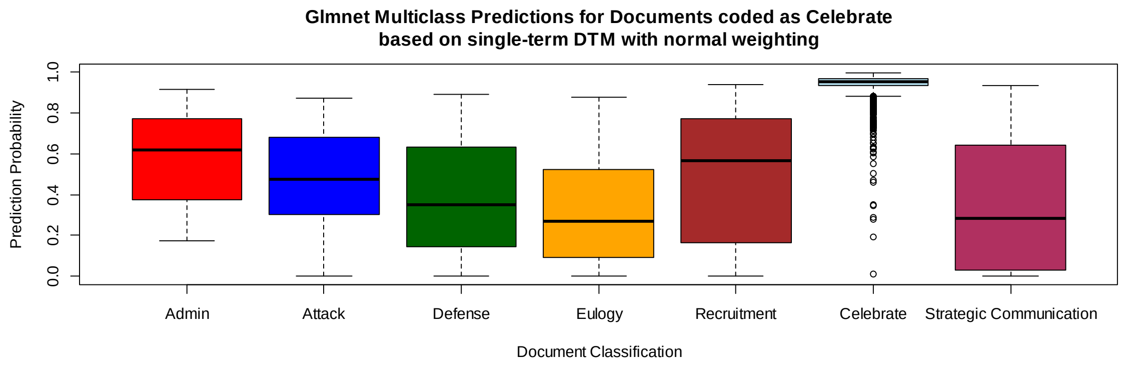
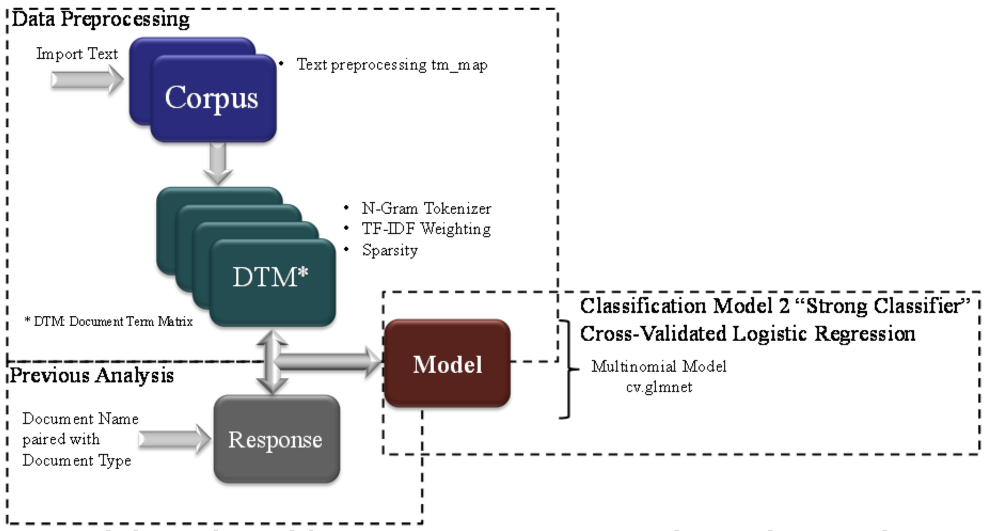


Figure 7. Model 1 Boxplots of the Estimated Probability that a Document is of Type "Celebrate" by Actual Document Type

2. Model 2: Strong Classifier

Data preprocessing of the corpus for Model 1 and Model 2 are identical. However, the DTM for Model 2 is subject to the transformations described in Chapter II. Here we use the DTM with additional bi-grams and tri-grams. In addition, we apply TF-IDF weighting and change the sparsity to 99 percent (Figure 8).


Data Preprocessing
Import Text
Corpus
Text preprocessing tm_map
DTM*
N-Gram Tokenizer
TF-IDF Weighting
Sparsity
* DTM: Document Term Matrix
Previous Analysis
Document Name paired with Document Type
Response
Model
Classification Model 2 "Strong Classifier"
Cross-Validated Logistic Regression
Multinomial Model
cv.glmnet

Preprocessing includes single-word, bi-grams, tri-grams, TD-IDF weighting and sparsity change to 0.99.

Figure 8. Diagram of Production Process for Classification Model 2.

The influence of TD-IDF and term tokenization to the tri-gram level provides a respectable increase in the cross-validated multinomial generalized linear model performance. For a small computational cost, the cross-validated misclassification rate reduces to 7.73 percent (Table 4). This model is the best performing multinomial classifier and is thus labeled as the "Strong Classifier."

Table 4. Model 2 Classification Table—Actual Versus Predicted

	Predicted Classification						
Coded Classification	**Admin**	**Attack**	**Celebrate**	**Defense**	**Eulogy**	**Recruitment**	**Strategic Comm**
Admin	0	0	14	0	0	0	0
Attack	0	7	30	2	0	0	11
Celebrate	0	0	2438	1	0	0	8
Defense	0	0	32	39	1	0	13
Eulogy	0	0	22	0	55	0	9
Recruitment	0	0	11	0	0	0	7
Strategic Communication	0	0	59	0	1	0	98

Cross-Validated Misclassification Rate: 7.73 percent.

The mean cross-validated misclassification rate over 100 runs for Model 2 is **7.195** percent. When we compare the predictive power of Model 2 with Model 1, Figure 9 shows that Model 2 is a stronger classifier of "Celebrate" documents.

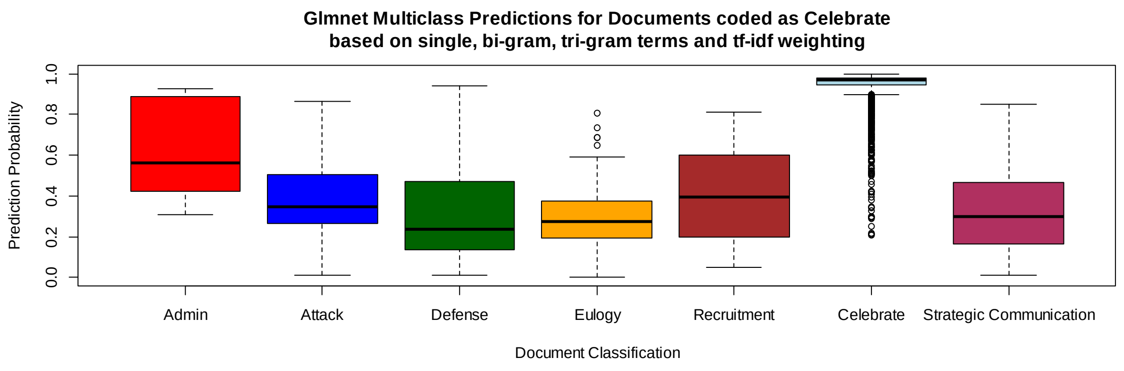


Figure 9. Model 2 Boxplots of the Estimated Probability that a Document is of type "Celebrate" by Actual Document Type.

C. CASCADE MODEL

In the pursuit of improving upon the performance of our classifiers, we consider the framework and methodologies demonstrated in research focused on real-time face detection. In the work of Viola and Jones (2014), a method is established "for combining classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions." Although slightly different in application, we consider parallels with the overwhelming number of "Celebrate" documents in the Islamic State Movement database.

One of the critical components to the Viola and Jones framework is the strategic use of different classifiers. Although our database of documents and features is not extremely large and computational speed is not a prohibitive factor, this will not always be the case. Like real-time face detection, an enormous corpus of text documents may prohibit training a traditional multinomial classification model. Viola and Jones (2004) explain that

> the key insight is that smaller, and therefore more efficient, boosted classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances. Simple classifiers are used to reject that majority of sub-windows before more complex classifiers are called upon to achieve low false positive rates.

The methodology for a detection cascade is displayed in Figure 10, where many easily-classified negative examples can be eliminated to reduce the number of observations that need further processing (Viola and Jones 2010). In a similar construct, we utilize our "Weak Classifier," Model 1, to determine with great confidence which documents are of type "Celebrate." After several iterations of classifying "Celebrate" documents and setting them aside, we bring the relative number of each document type in a new DTM into balance. This subsequently enables us to use the "Strong Classifier," Model 2, to finalize the classification process. By keeping track of the classifications at every level of the cascade, it is possible to reassemble the results to classify the entire corpus.

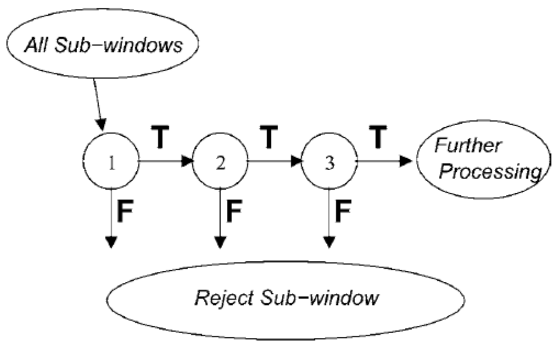


Figure 6. Schematic depiction of a the detection cascade. A series of classifiers are applied to every sub-window. The initial classifier eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation. After several stages of processing the number of sub-windows have been reduced radically. Further processing can take any form such as additional stages of the cascade (as in our detection system) or an alternative detection system.

Figure 10. Face Detection Cascade Model Schematic.
Source: Viola and Jones (2004).

As with Model 1 and Model 2, the cascade model construct first preprocesses the text data into a corpus followed by a DTM. Because both the "Weak Classifier" and "Strong Classifier" are used in the cascade, we prepare two distinct DTMs. Figure 11 shows the addition of the layered classifier. First, the cascade model uses a binomial cross-validated classification model to determine if a document is a "Celebrate" or a "Not Celebrate" document. Rather than classifying documents as "Celebrate" if the estimated probability of "Celebrate" is greater than 0.50, we raise the threshold to 0.94. This identifies approximately 1000 documents as type "Celebrate" while providing zero false positives (documents classified as "Celebrate" which actually belong to one of the other six document types). False negatives are allowed at this stage. This process is repeated two additional times with decreasing thresholds of 0.84 and 0.70 respectively. At this point in the cascade, over 2000 "Celebrate" documents have been properly classified and set aside, leaving a more balanced number of each type of document in the DTM (Figure 12).

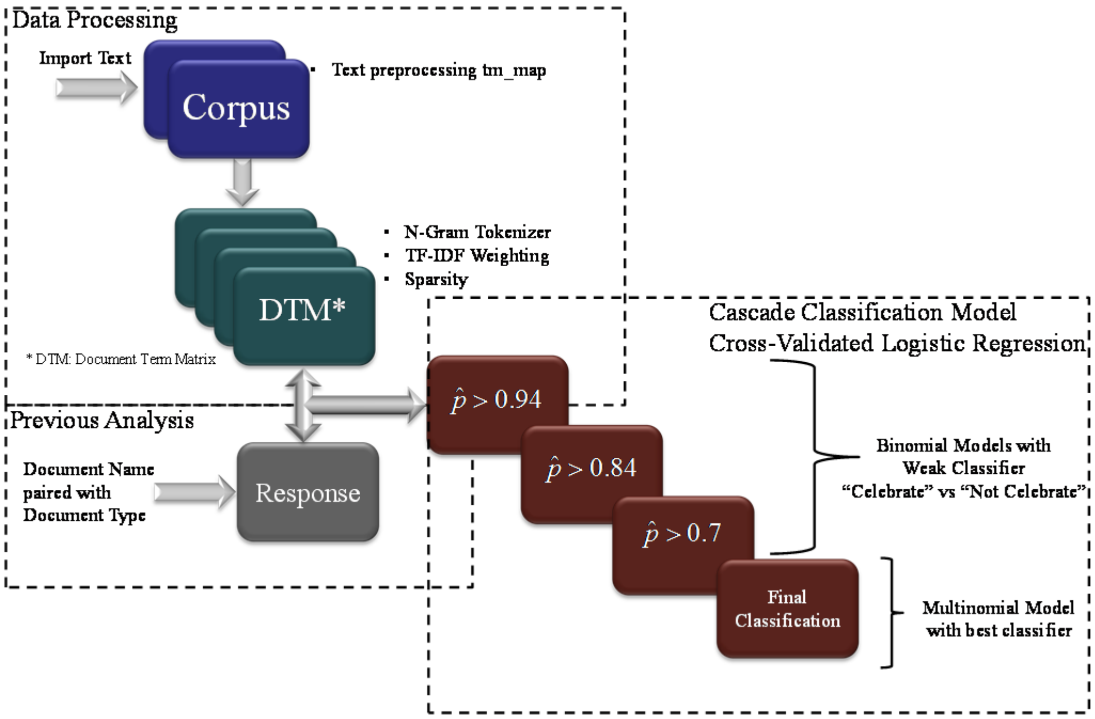

Data Processing
Import Text
Corpus
Text preprocessing tm_map
N-Gram Tokenizer
TF-IDF Weighting
Sparsity
DTM*
* DTM: Document Term Matrix
Previous Analysis
Document Name paired with Document Type
Response
Cascade Classification Model
Cross-Validated Logistic Regression
$\hat{p} > 0.94$
$\hat{p} > 0.84$
$\hat{p} > 0.7$
Final Classification
Binomial Models with Weak Classifier "Celebrate" vs "Not Celebrate"
Multinomial Model with best classifier

Figure 11. Diagram of Production Process for Cascade Classification Model.

```
       Admin      Attack        Defense    Eulogy
          14          50             85        86
 Recruitment   Celebrate  Strategic Comm
          18         175            158
```

Figure 12. Approximate Document Type Distribution Upon Reaching the Fourth Level of the Cascade Classification Model.

Once we reach the fourth level of the cascade model, the classifier is replaced with a multinomial "Strong Classifier" applied to a database with more evenly distributed document types. Upon receiving the results of the multinomial classification, we reassemble the overall predicted classes from each level of the cascade. An example of the final classification results of the cascade method are shown in Table 5.

Table 5. Cascade Model Classification Table—Actual Versus Predicted

	Predicted Classification						
Coded Classification	**Admin**	**Attack**	**Celebrate**	**Defense**	**Eulogy**	**Recruitment**	**Strategic Comm**
Admin	2	0	6	0	1	0	5
Attack	0	17	6	3	0	0	24
Celebrate	0	0	2419	3	2	0	23
Defense	0	0	14	54	2	0	15
Eulogy	0	0	2	0	70	0	14
Recruitment	0	0	1	0	2	1	14
Strategic Communication	0	0	19	1	1	0	137

Cross-Validated Misclassification Rate: 5.5 percent.

Although the cascade model still faces challenges with a database containing some rare document types, it overwhelming outperforms Model 1 and Model 2. The example shown in Table 5 demonstrates a 5.5 percent cross-validated misclassification rate, which surpasses the 9 percent and 7.7 percent cross-validated misclassification rates shown in Table 3 and Table 4. The mean cross-validated misclassification rate over 100 runs of is **5.71** percent.

The cascade model attempts to compensate for characteristics present in many databases, mainly the limited size of the number of observations and the unbalanced distribution of the categorical response variable. We assume that the distribution of document types for a new set of documents sourced from Islamic State Movement officially sanctioned media outlets will be similar. This means that roughly 80 to 90 percent of the documents in a future corpus will be "Celebrate" documents. We feel this assumption is reasonable and the model thresholds could be "tweaked" to accommodate for any significant differences.

IV. DATA VISUALIZATION

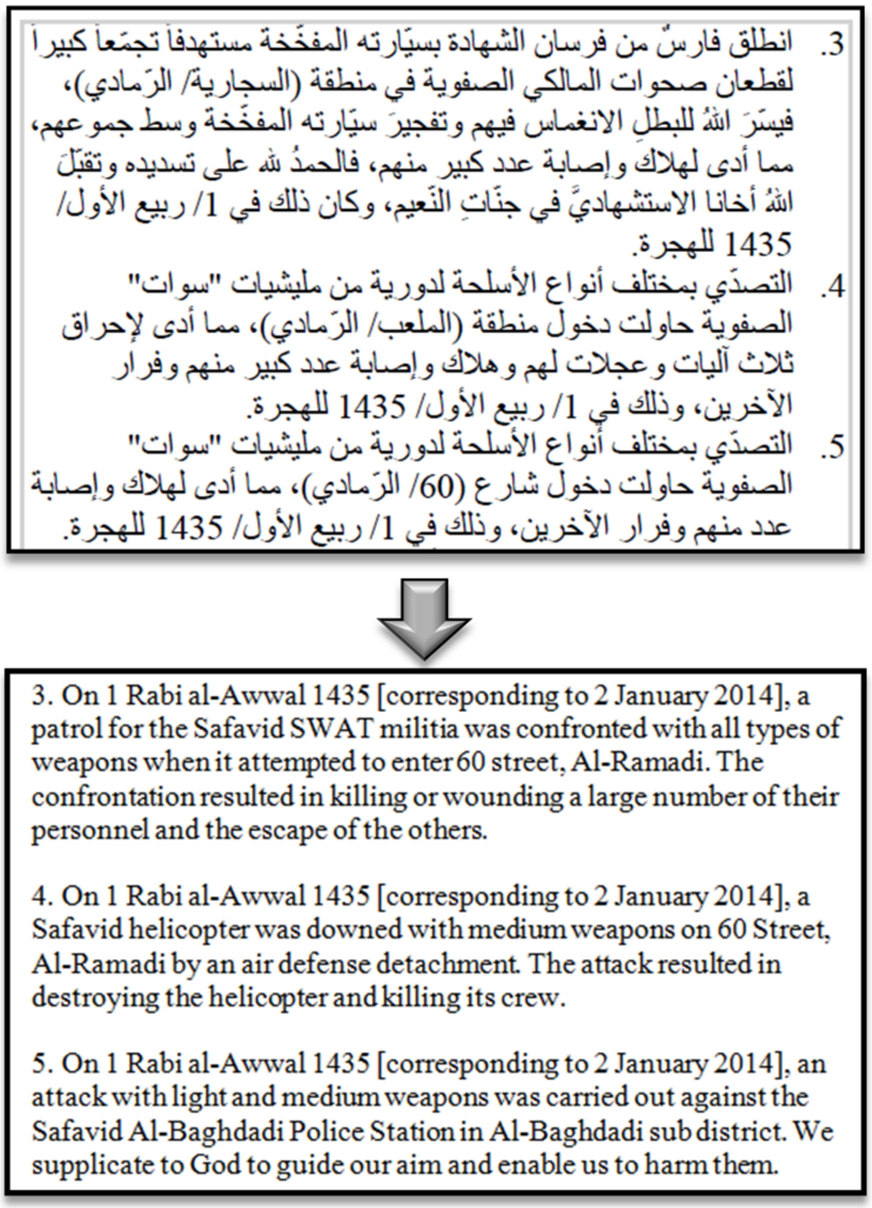
A. VISUALIZATION TOOL

The Islamic State Movement “Celebrate” documents exhibit a standardized and highly structured format. These documents are rich with data that has been meticulously collected and published by its officially sanctioned media outlets. Each operational summary contains a greeting and an introduction to the general content of the document, followed by a body of specific events, and a conclusion which often declares praise to God for their accomplishments. This unique structure proves to be highly advantageous as we develop an analytic tool to visualize and understand the wealth of information contained within the summaries.

The visualization of the data within the Islamic State “Celebrate” documents provides a much deeper understanding of the Islamic State battle plan and strategy. With the ability to see the geographic relationships of events as they relate to time and space, analysts are able to deepen their understanding of history and the methods by which the Islamic State operated in Iraq over a period of ten years. Identifying patterns and trends can help us understand the Islamic State as they conduct their current operations.

B. DATA EXTRACTION

The focus of our data visualization efforts is to extract and display the specific events contained in the body of each “Celebrate” document. The descriptive accounts of each event claimed by the Islamic State Movement provide the date and location of each attack, occasionally the time of day, as well as a description of the attack and its target. Many of the events are numbered and we leverage this characteristic as we develop our tool. Figure 13 shows the structure of the body of a typical “Celebrate” document in its original Arabic format, as well as its preserved structure translated into English. Occasionally these event descriptions start with the word “On” followed by the day of the week. We account for both formats in the event search methodology.



3. انطلق فارسٌ من فرسان الشهادة بسيّارته المفخّخة مستهدفاً تجمّعاً كبيراً لقطعان صحوات المالكي الصفوية في منطقة (السجارية/ الرّمادي)، فيسّرَ اللهُ للبطلِ الانغماس فيهم وتفجيرَ سيّارته المفخّخة وسط جموعهم، مما أدى لهلاك وإصابة عدد كبير منهم، فالحمدُ لله على تسديده وتقبّلَ اللهُ أخانا الاستشهاديَّ في جنّاتِ النّعيم، وكان ذلك في 1/ ربيع الأول/ 1435 للهجرة.
4. التصدّي بمختلف أنواع الأسلحة لدورية من مليشيات "سوات" الصفوية حاولت دخول منطقة (الملعب/ الرّمادي)، مما أدى لإحراق ثلاث آليات وعجلات لهم وهلاك وإصابة عدد كبير منهم وفرار الآخرين، وذلك في 1/ ربيع الأول/ 1435 للهجرة.
5. التصدّي بمختلف أنواع الأسلحة لدورية من مليشيات "سوات" الصفوية حاولت دخول شارع (60/ الرّمادي)، مما أدى لهلاك وإصابة عدد منهم وفرار الآخرين، وذلك في 1/ ربيع الأول/ 1435 للهجرة.

3. On 1 Rabi al-Awwal 1435 [corresponding to 2 January 2014], a patrol for the Safavid SWAT militia was confronted with all types of weapons when it attempted to enter 60 street, Al-Ramadi. The confrontation resulted in killing or wounding a large number of their personnel and the escape of the others.

4. On 1 Rabi al-Awwal 1435 [corresponding to 2 January 2014], a Safavid helicopter was downed with medium weapons on 60 Street, Al-Ramadi by an air defense detachment. The attack resulted in destroying the helicopter and killing its crew.

5. On 1 Rabi al-Awwal 1435 [corresponding to 2 January 2014], an attack with light and medium weapons was carried out against the Safavid Al-Baghdadi Police Station in Al-Baghdadi sub district. We supplicate to God to guide our aim and enable us to harm them.

Figure 13. "Celebrate" Document Common Format with Numbered Events Adapted from Hanin Network Forums (2014)

The first objective in the data visualization process is to extract and organize the information contained in the body of each operational summary. The goal for this objective is to create an **R** data set called a data frame where each row is an event that contains the document name from which it came, the raw text from the event, and the day, time, location, and latitude and longitude for each event.

1. Events

In order to create the data frame, we start with the original corpus of documents. We employ two functions, shown in Appendix E, which utilize regular expressions to extract only those lines of text that correspond to events. The text corresponding to these events are preserved in the data frame, while all document text contained in the introduction and conclusion of the document are discarded. We pair each event, which has its own row, with its document name for future reference. After processing all documents, the data frame contains 13,798 rows, corresponding to the total number of events. A small percentage of the "Celebrate" documents do not adhere to the standardized formats and therefore we are unable to include these events in our final tool.

2. Dates

The data frame now contains text that corresponds to the descriptive account of each event. The challenge is to extract key pieces of information from the text so that they can be accurately categorized as a date, time, or location. Extracting the date and time of each event is relatively straightforward. To extract the dates, we use the `str_extract()` function from the **stringr** package (Wickham 2015). For example, the data frame of events provides the following text:

```
"2. At 1100 on Wednesday, 1 Dhu al-Qi'dah 1427, correspondi
ng to 22 November 2006, one of the Crusader snipers was sho
t at the Crusader army headquarters in Hit. Praise and grat
itude be to God."
```

The `find.date()` function, shown in Appendix F, relies on the structure of the translated English date, but it can also accommodate one or two digit dates as well as abbreviated or full spellings of the month. The function outputs the following string:

```
"22 November 2006"
```

We subsequently convert the date string with the `strptime()`function, a base **R** function, from a character string to a "POSIXlt" class object with the appropriate Iraq time zone, yielding the date `“2006-11-22 ADT.”` This date format is provided to the user in the final data visualization tool.

3. Times

A similar function utilizes regular expressions to capitalize on the consistent format of presenting the times of the events. The `find.time()` function searches for the pattern of the word “at” followed by four digits, where “at” can include any combination of upper or lower-case letters. Applied to the same string describing the November 2006 attack, the `find.time()` function returns `“1100.”` The data frame now contains the following columns:

```
“docname” “text” “dates” “times”
```

4. Location

The next objective is to extract a location from each attack event in the data frame. One assumption we make is that extracted event only describes an event that occurs in one geographic location. If the event described, for example, highlights a series of simultaneous attacks in several locations, the algorithm will not capture all locations. Secondly, the algorithm we use is not foolproof. It makes assumptions to mechanically maximize the accuracy and fidelity of the extracted location, and thus it will not accommodate every kind of text presentation of a location description.

Through analyzing the structure of the “Celebrate” documents, we observe that there are patterns in the presentation of locations within the text. For example, a common string of text contains the following:

```
“17. On 18 Dhu-al-Qi’dah 1431 [corresponding to 26 October
2010], one of the apostate spies was targeted with the
detonation of a sticky explosive device on his vehicle in
Al-Sihhah neighborhood in Al-Dawrah area. The explosion
resulted in seriously wounding him.”
```

This event presents both a neighborhood and an area. Our algorithm attempts to account for these differences in fidelity and extract the “neighborhood” because it will logically have a more exact location. Another example includes:

```
“6. On Saturday, 8 Dhu-al-Qi’dah 1428, corresponding to 17
November 2007, an explosive was detonated against a four-
wheel drive vehicle belonging to the Coalition forces on
Al-Mu’askar Street, in Al-Rastamiyah area, destroying it
and killing or wounding everyone inside it. Praise and
gratitude be to God.”
```

In this attack, there is a street name provided which we desire to prioritize over the area location. However, we must also prepare for the possibility that we may not be able to locate a particular street name on a map.

In order to maximize the accuracy and fidelity of the extracted locations, we create five search categories and prioritize them. The search categories are listed in decreasing priority in Table 6. We utilize regular expressions within the **R** code to extract strings that match exact patterns. In the process, we accommodate situations where we face boundaries, such as the presence of punctuation at the beginning or end of a string. With the search functions, we also remove the word “the” from the function outputs in order to increase the chance of matching a location name.

For the first four location search categories, we use regular expressions to extract location names that precede the words “street” or “road,” “neighborhood,” “district,” or “area.”

Table 6. Prioritized Location Search Categories, Patterns, and Functions

Search Category	String search pattern, ignoring case	Function
street or road	Between "in" or "on" and "street" or "road"	find.road()
neighborhood	Between "in" and "neighborhood"	find.neigh()
district	Between "in" and "district"	find.district()
area	Between "in" and " area"	find.area()
misc	Between "in" and next punctuation **	find.misc()
** For the misc category, the following punctuation has previously been removed by the rm_punc () function: '" 's "," ' ", " "," - "		

The `find.misc()` function is the most complex use of regular expressions and accommodates more than one pattern match within each event. It extracts strings bracketed by the word "in" and a comma, period, or the end of the event text. After the function extracts these strings, regular expressions further refine the misc category solution by searching for key words in the string and removing words like "the" which prevent a location match in subsequent steps. Observe the following string:

```
"The attack took place in Mosul, in the center of the
street, in the vicinity of the Yaqten Restaurant . Praise
be to God."
```

The `find.misc()` function, shown in Appendix G, applied to this string returns the following results:

```
"Mosul" "street" "Yaqten Restaurant"
```

By utilizing all five search categories we attempt to maximize the number of events with an assigned location. Despite this layered approach, our algorithm is unable to assign a location for every event. Some attacks simply do not provide a location in their account. However, many other accounts present unique challenges related to language and text that cause our methodology to fall short. Therefore, we develop an additional tool to help increase the likelihood of assigning a location to each event. To accomplish this, we use a search through a location database. We will first introduce the database and its necessity, and subsequently describe the search technique as it fits within our overall search algorithm.

C. GEONAME LOCATION DATABASE

Extracting the locations is a critical component of creating a tool to visualize the data contained within each of the events claimed by the Islamic State Movement. However, in order to build the tool, we must be able to plot each attack's location to an exact location on a map. There are many methods to achieve this end, but due to the high geographic fidelity of many of the attack accounts, we choose to cross-reference an Iraq database that contains latitudes and longitudes.

The GeoNames geographical database is an unclassified publically available database that can be downloaded under a creative commons attribution license. The database "contains over 10 million geographical names and consists of over 9 million unique features whereof 2.8 million populated places and 5.5 million alternate names" (GeoNames 2016). The locations are categorized into one of nine feature classes and further subdivided into 645 feature codes, which include categories such as cities, districts, areas, streets, parks, buildings, and lakes (GeoNames 2016).

The current Iraq database contains 31,894 locations. These primary locations also collectively maintain 162,851 alternate names. In order to avoid text processing problems, we delete non-ASCII characters, which removes all Arabic alternate names while preserving those in the English language. After removal, our database contains 79,263 alternate names. A critical attribute of the GeoNames database is that each location has an assigned latitude and longitude in the World Geodetic System 1984 (WGS84) format. In **R**, we construct a new data frame that contains three columns. The first column contains all primary and alternate names. The second and third columns contain latitudes and longitudes corresponding to all 111,157 location names. This data frame allows us to match location names to events within the "Celebrate" documents and assign matched events with a latitude and longitude.

D. LOCATION MATCHING

The two greatest obstacles in building a data visualization tool are extracting location names from the text and the limited scope of most Iraq place name databases. Although the GeoNames database is relatively comprehensive, it is difficult to match all manners of descriptions locations. Some of these challenges arise when presented with a pseudonym or generic name to represent a particular location, such as the use of the phrase “capital city” to represent Baghdad or phrases such as “northern Baghdad.” Our algorithms attempt to accommodate some of these presentations, but without the use of machine learning techniques, for example, we face limitations.

In order to prioritize location selections and maximize the ability to plot these locations, our algorithm systematically descends the search categories shown in Table 6. For example, if our functions extract a neighborhood name and an area name from an event text, the algorithm prioritizes the neighborhood name due to its fidelity, but only if the neighborhood name contains coordinates in the GeoName database. If only the area name provides matched coordinates, the area name is selected.

The first four search categories are the most accurate in extracting location names likely to be found in the GeoName database. The misc category is highly beneficial but less accurate due to its relaxed search design. By using the first four categories, 57.4 percent of the events are assigned location names. By adding the misc search category over 90 percent of the events contain a string to represent the location name.

To enhance accuracy and improve the ability to assign coordinates to each event, we develop an additional search method. This process takes each of the 111,157 location names within the GeoName database and attempts to locate them within each of the 13,798 events. This method is computationally expensive and requires over 1.53 billion searches. To address these requirements, we take advantage of the increased capacity provided by the Naval Postgraduate School’s (NPS) High Performance Computing (HPC) resources. The Hamming supercomputer cluster enables us to leverage 64 cores of computing power and reduce a 44-hour search on a four-core laptop to two hours and 45 minutes. We run the Hamming job with an **R** script file (Appendix H). The search finds 6,100 location matches within the text, of which 1,138 reside within unique events. The search increases the number of events with coordinates by 768, providing a 20 percent increase in the number of events available for plotting.

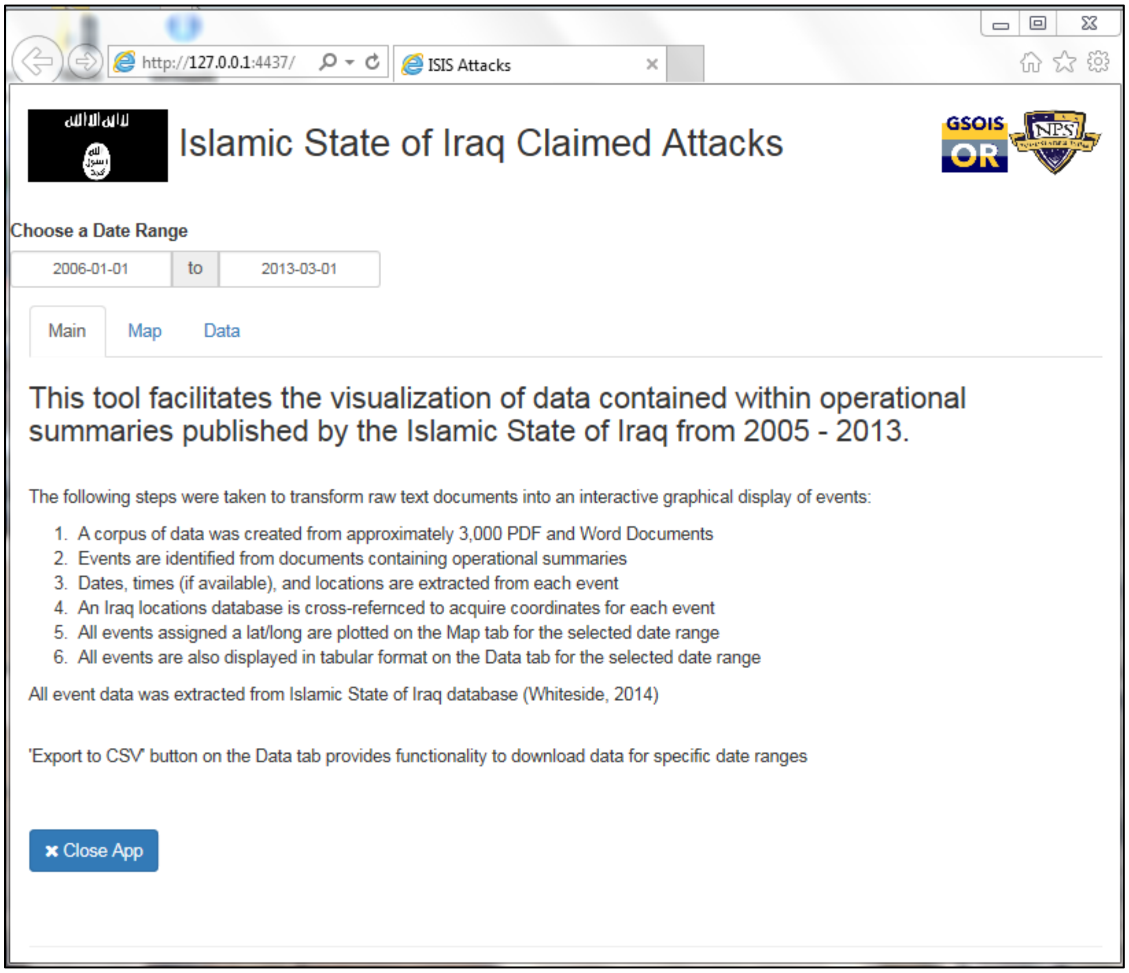
The misc search category is prioritized last in the location acquisition logic. Approximately 45 percent of the locations extracted by the misc search category do not result in matching coordinates. However, 2,204 of its findings are used to plot Islamic State Movement events.

By combining and prioritizing all location search methods, our algorithm provides location names for 91.2 percent of the 13,798 total events and coordinates for 42 percent of those events.

E. R SHINY VISUALIZATION TOOL

In order to provide the user with a tool to visualize each event, the data must be consolidated and formatted. We construct a final data frame that contains the following columns: event number, document name, event text, date, time, location name, latitude, and longitude. The information also serves to help the user understand the data once in its visual format. We construct the tool to visualize this data with two main **R** packages, **Shiny** and **leaflet**. The **leaflet** package is used within **Shiny** to provide an interface to JavaScript libraries to access interactive maps (Cheng and Xie 2016). It provides functionality to develop pop-up markers and cluster items plotted on a large variety of map backgrounds. The JavaScript behind the package provides the ability to zoom in to neighborhoods and maintains scroll functionality typically found in web-interactive applications. The **Shiny** package provides the ability to develop an interactive tool in **R** that can be hosted by a local computer or a server to provide numerous users with an interactive web-based tool (Chang, Cheng, Allaire, Xie, McPherson 2016). We develop a user interface and a server within **Shiny** which provides the **R** code needed to host the data visualization tool.

We format the tool with three tabs and functionality to select a date range. The Main tab, seen in Figure 14, provides background information and basic instructions. The user can close the application from the Main tab.



Islamic State of Iraq Claimed Attacks

Choose a Date Range

2006-01-01 to 2013-03-01

This tool facilitates the visualization of data contained within operational summaries published by the Islamic State of Iraq from 2005 - 2013.

The following steps were taken to transform raw text documents into an interactive graphical display of events:

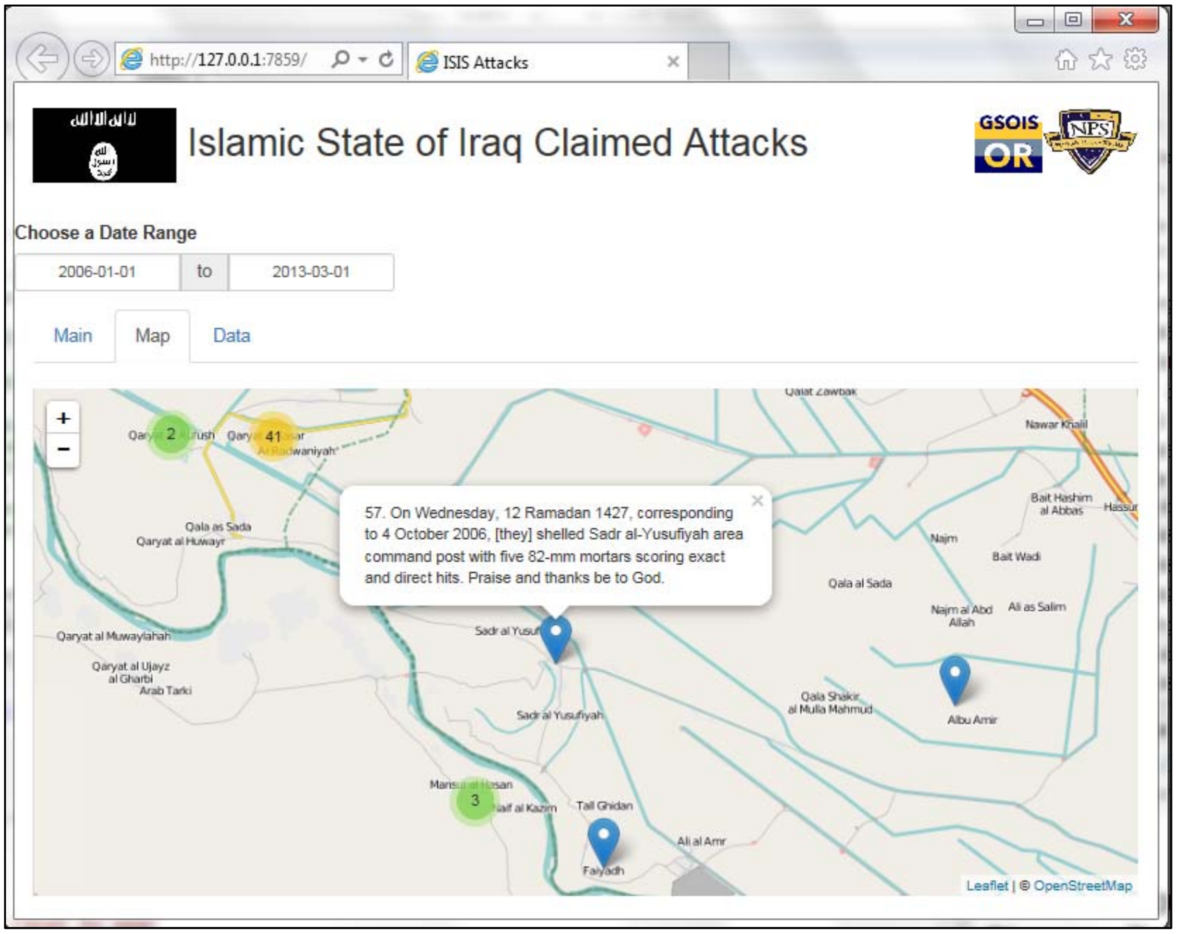
1. A corpus of data was created from approximately 3,000 PDF and Word Documents
2. Events are identified from documents containing operational summaries
3. Dates, times (if available), and locations are extracted from each event
4. An Iraq locations database is cross-refernced to acquire coordinates for each event
5. All events assigned a lat/long are plotted on the Map tab for the selected date range
6. All events are also displayed in tabular format on the Data tab for the selected date range

All event data was extracted from Islamic State of Iraq database (Whiteside, 2014)

'Export to CSV' button on the Data tab provides functionality to download data for specific date ranges

Figure 14. Main Tab for **R Shiny** Interactive Tool

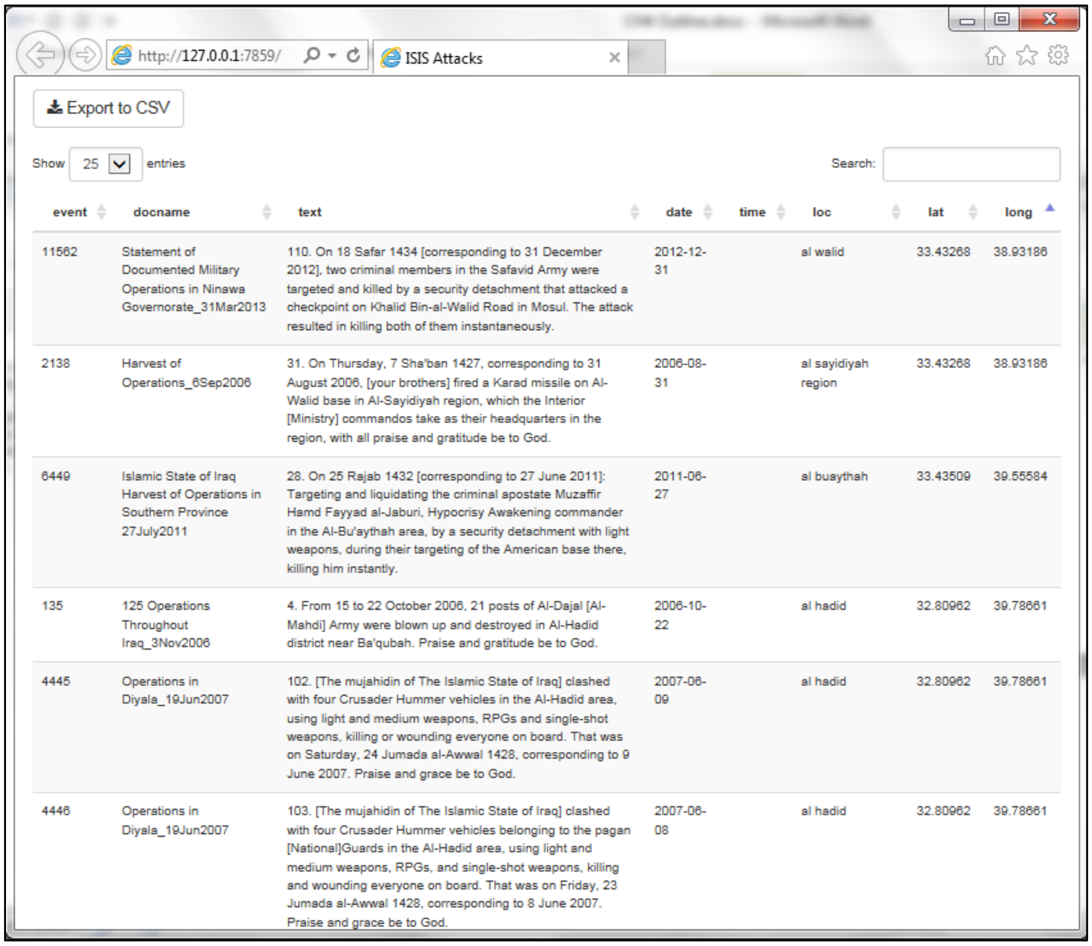
The Map tab, shown in Figure 15, contains the interactive map with the plotted events for the date range selected. As the user zooms in, the events separate from their clusters and each attack is represented by a blue marker. By clicking on a marker, the text from that event displays. Figure 15 shows an Islamic State Movement account of a mortar attack from October 2006, which occurred southwest of Baghdad in the Sadr Al-Yusufiyah area.



57. On Wednesday, 12 Ramadan 1427, corresponding to 4 October 2006, [they] shelled Sadr al-Yusufiyah area command post with five 82-mm mortars scoring exact and direct hits. Praise and thanks be to God.

Figure 15. Map Tab for **R Shiny** Interactive Tool

The Data tab, shown in Figure 16, displays the data frame that generates the event markers on the Map tab. The data frame is searchable and can be sorted by column or filtered by date range. An "Export to CSV" button allows the user to download and save information from their custom search or filter.



event	docname	text	date	time	loc	lat	long
11562	Statement of Documented Military Operations in Ninawa Governorate_31Mar2013	110. On 18 Safar 1434 [corresponding to 31 December 2012], two criminal members in the Safavid Army were targeted and killed by a security detachment that attacked a checkpoint on Khalid Bin-al-Walid Road in Mosul. The attack resulted in killing both of them instantaneously.	2012-12-31		al walid	33.43268	38.93186
2138	Harvest of Operations_6Sep2006	31. On Thursday, 7 Sha'ban 1427, corresponding to 31 August 2006, [your brothers] fired a Karad missile on Al-Walid base in Al-Sayidiyah region, which the Interior [Ministry] commandos take as their headquarters in the region, with all praise and gratitude be to God.	2006-08-31		al sayidiyah region	33.43268	38.93186
6449	Islamic State of Iraq Harvest of Operations in Southern Province 27July2011	28. On 25 Rajab 1432 [corresponding to 27 June 2011]: Targeting and liquidating the criminal apostate Muzaffir Hamd Fayyad al-Jaburi, Hypocrisy Awakening commander in the Al-Bu'aythah area, by a security detachment with light weapons, during their targeting of the American base there, killing him instantly.	2011-06-27		al buaythah	33.43509	39.55584
135	125 Operations Throughout Iraq_3Nov2006	4. From 15 to 22 October 2006, 21 posts of Al-Dajal [Al-Mahdi] Army were blown up and destroyed in Al-Hadid district near Ba'qubah. Praise and gratitude be to God.	2006-10-22		al hadid	32.80962	39.78661
4445	Operations in Diyala_19Jun2007	102. [The mujahidin of The Islamic State of Iraq] clashed with four Crusader Hummer vehicles in the Al-Hadid area, using light and medium weapons, RPGs and single-shot weapons, killing or wounding everyone on board. That was on Saturday, 24 Jumada al-Awwal 1428, corresponding to 9 June 2007. Praise and grace be to God.	2007-06-09		al hadid	32.80962	39.78661
4446	Operations in Diyala_19Jun2007	103. [The mujahidin of The Islamic State of Iraq] clashed with four Crusader Hummer vehicles belonging to the pagan [National]Guards in the Al-Hadid area, using light and medium weapons, RPGs, and single-shot weapons, killing and wounding everyone on board. That was on Friday, 23 Jumada al-Awwal 1428, corresponding to 8 June 2007. Praise and grace be to God.	2007-06-08		al hadid	32.80962	39.78661

Figure 16. Data Tab for **R Shiny** Interactive Tool

The Islamic State Movement data visualization tool provides the user with the ability to analyze trends in attacks and more effectively see relationships between time and space for the data contained within thousands of text documents. The algorithms and methodologies we use to produce this tool are not perfect and the next chapter will discuss potential ways to improve and expand upon the tool created in this thesis.

V. SUMMARY AND FUTURE WORK

A. SUMMARY

The Islamic State Movement database collected by Whiteside (2014) holds to a principled collection methodology and provides a pure and unique point of view based on its sources. Analysis of the database's 2,858 documents creates a burdensome challenge for manual methods. With the application of **R** statistical software, and its **tm** and **glmnet** packages in particular, we are able to alleviate some of these challenges by applying NLP tools. We leverage Whiteside's manual document analysis to develop supervised cross-validated classification models and generate the capability to automatically classify Islamic State Movement documents into one of seven types based only on the terms present in the text of a document. To address modeling challenges related to an unbalanced distribution of document types, we develop a cascade cross-validated **glmnet** classification model, which provides the best performance of all models with a mean misclassification rate of 5.71 percent.

Over 85 percent of the Islamic State Movement documents within Whiteside's database are classified as type "Celebrate." These operational summaries provide an opportunity to develop an interactive tool to help the user visualize the rich data within the text. By developing a user friendly and highly interactive **R Shiny** tool, users can see and explore their data based on geographic and time relationships, enabling easier and more effective trend analysis. The ability to plot the event locations is dependent on our algorithm's ability to extract locations and match these locations to an open-source GeoName database. Our visualization tool provides location recommendations for 91.2 percent of the 13,798 total events extracted from the database, of which 42 percent of the events are plotted.

B. FUTURE WORK

The body of work provided in this thesis provides some techniques and methodologies for alleviating the burden of manual text processing. The document classification model and data visualization platform provide two helpful tools in contributing to this goal. Other helpful tools could be developed to augment those developed in this thesis. Also, there is room to improve on our tools and methodologies. Several improvements for future work include:

1. Collect a Test Set

With the size of Whiteside's Islamic State Movement database, we chose to develop a cross-validated classification model. Future work should consist of collecting a test set which includes documents from 2014 until the present. This new corpus can be used as a test set to validate our classification models. Special attention should be paid to understanding the authenticity of the document sources while collecting a test set, as the Islamic State's officially sanctioned media outlets are continuously evolving. We conducted some research in this area and there is evidence to believe that the structure of the "Celebrate" documents remains consistent in current written documentation.

2. Web-Scraping Functionality

In addition to manually collecting a more current test set for analysis, future work could include developing a web-scraping tool to collect press releases from official Islamic State media outlets and provide periodic updates to corpus. This tool could include a periodic refresh cycle that provides a nearly real-time update to the corpus of documents.

3. Classify by Type of Attack

The **R Shiny** data visualization tool could be expanded to identify the type of attack within each event, such as "IED," "mortar attack," or "suicide bomb." Similar to our document type classification models, NLP tools could be applied to classify events according to the type of attack.

4. Comprehensive R Shiny Tool

The data visualization tool could be expanded to include the full body of work developed in this thesis. By creating functionality within the **R Shiny** platform to perform all operations, beginning with uploading documents of various formats to create a corpus, the entire process could be self-contained and executable by any user.

5. Improve Location Identification with Learning

Improvements are needed to comprehensively and reliably extract location information from each event. The algorithms we implement rigidly make assumptions about the text formatting and location presentation. Our methods provide a "best approximation" of the location, and although over 91 percent of the events are assigned a string in the location category, some of these locations are erroneous. Our algorithms lack total reliability and future work could include the ability to "learn" location names. An approach to accomplish this is to use the events with location names that have corresponding coordinates as a training set and then use supervised machine learning methods or NLP methods to train models to predict locations from the document text based on this training set.

6. Updating and Improving the Location Database

Updates to the visualization tool could include creating functionality to automatically refresh the Iraq GeoName database within the **R Shiny** interface or replacing it with a more comprehensive database.

7. Editing the Data Tab Data Frame

Improvements to the visualization tool could include providing the user with the ability to edit the data frame resident in the **R Shiny** tool's Data tab in order to update event locations. This functionality could provide the user with the ability to select an individual event and then choose a location from a drop-down list or manually enter a new location or set of coordinates.

8. Expanding to New Media Types

By expanding the scope of this thesis and Whiteside's database, the future of analysis related to the Islamic State Movement should not be confined to written media.

As Winter (2015) describes in his research titled "Documenting the Virtual 'Caliphate,'" the group is transitioning away from written text and traditional forums.

> These forums still exist and are still used; however, they now play a role secondary to that of open source (Twitter, Tumblr, Facebook etc.) and peer-to-peer (Kik, Surespot, Telegram etc.) social media, as jihadists have become less insular and have sought to increase their exposure and accessibility (Winter 2015).

Future work should attempt to include social media inputs, specifically Twitter, to include photo and video media, as well as text. As Winter (2015) illustrates in Figure 17, 78 percent of the Islamic State's propaganda is distributed in the form of photos. This opens up an entirely new area of research, which includes automated image and video recognition to help analysts identify events and locations. Machine learning such as neural networks techniques will prove to be highly advantageous in this area of expanding popularity.

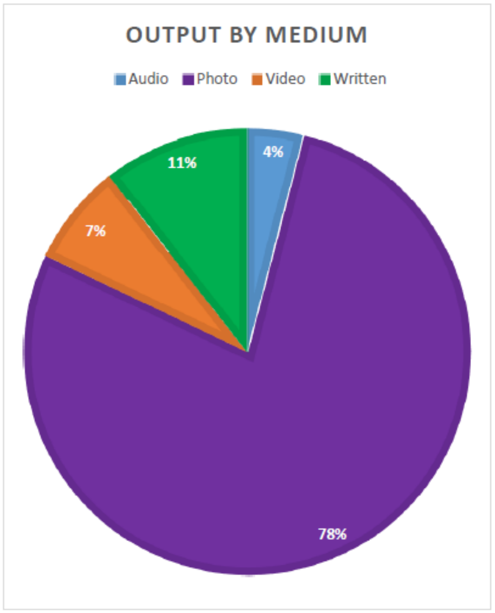


Figure 17. Islamic State Output by Media Source: Winter (2015)

APPENDIX: R FUNCTIONS AND SCRIPTS

A. DOCX TO TEXT CONVERSION

```
# Document conversion from docx format to txt format
# Download docx2txt.tgz
# https://sourceforge.net/projects/docx2txt/?source=typ_redirect
# Install and configure
# Executable location c:\tools\docx2txt\docx2txt.bat

exec <- “c:/tools/docx2txt/docx2txt.bat”

# For each directory you desire to convert, list the .docx files
there
base <- “P:/”
subdir <- “2006”
mydir <- paste0 (base, subdir)
setwd (mydir)

# Find documents with the docx file extension
docx <- list.files (pattern=“\\.docx$”)

# Remove the docx file extension
basename <- substring (docx, 1, nchar (docx) - 5)

# Give all documents a txt file extension
txt <- paste0 (basename, .”txt”)

# Stopping criteria
if (any (is.element (txt, list.files()))) cat (“Stop!\n”)

# Loop through all documents and convert to txt files
for (i in docx) {
cmd <- paste0 (exec, “ \,”“ i, “\”“)
system (cmd)
}
```

B. PDF TO TEXT CONVERSION

```
# Document Conversion from PDF to txt format
# Download Xpdf free open source PDF viewer/converter
# http://www.foolabs.com/xpdf/download.html
# Unpack tar.gz file

# Set directory for PDF conversion
dir <- “P:/2004”

# Make a vector of PDF file names
myfiles <- list.files(path = dir, pattern = “unlocked.pdf,”
     full.names = TRUE)

# Create a txt file for every pdf in the directory
lapply(myfiles,function(i) system(paste(‘“C:/xpdf/xpdfbin-win-
     3.04/bin64/pdftotext.exe”‘,paste0(‘“‘,  i,  ‘“‘)),  wait  =
     FALSE))
```

C. REMOVEMOSTPUNCTUATION FUNCTION

```
removeMostPunctuation<-
function (x, preserve_intra_word_dashes = FALSE)
{
rmpunct <- function(x) {
x <- gsub(,”‘“ “\002,” x)
x <- gsub(“[[:punct:]]+,” ,”“ x)
gsub(“\002,” ,”‘“ x, fixed = TRUE)
}
if (preserve_intra_word_dashes) {
x <- gsub (“([[:alpha:]])-([[:alpha:]]),” “\\1\001\\2,” x)
x <- rmpunct(x)
gsub(“\001,” “-,” x, fixed = TRUE)
} else {
rmpunct(x)
}
}
# Adapted from http://stackoverflow.com/questions/27951377/tm-
removepunctuation-except-hashtag
```

D. TM_MAP ORDERED PREPROCESSING ON CORPUS

```
myCorpus <- tm_map(Corpus, content_transformer(tolower))
myCorpus <- tm_map(myCorpus,removeNumbers)
myCorpus <- tm_map(myCorpus,removeWords,stopwords(“SMART”))
myCorpus <- tm_map(myCorpus,content_transformer(remove
           MostPunctuation), preserve_intra_word_dashes = TRUE)
myCorpus <- tm_map(myCorpus,stemDocument)
myCorpus <- tm_map(myCorpus,stripWhitespace)
```

E. FUNCTIONS TO EXTRACT EVENTS

```
# This function extracts events that start a string
# with a number 1–999 followed by a period. Ex: “152. On.”

check.start.num <- function(strg){
grepl(“^[1-9][0-9]{0,2}\\.,”trim(strg))
}
# This function extracts events that start a string with
# the word “On” followed by a space and then the day of the #
week. Ex: “On Friday, 18..”

check.start.on <- function(strg){
grepl(“^On\\sMonday|^On\\sTuesday|^On\\sWednesday|^On\\sThursday|
^On\\sFriday|^On\\sSaturday|^On\\sSunday,”trim(strg))
}

# Both functions above also utilize the trim function which #
removes any whitespace from the start and end of a string

trim <- function (x) gsub(“^\\s+|\\s+$,” “,” x)
```

F. FIND DATE FUNCTION

```
find.date <- function(strg){str_extract(strg,pattern =
“\\d{1,2}\\s\\b(?:Jan(?:uary)?|Feb(?:ruary)?|Mar(?:ch)?|Apr(?:il)
?|May|Jun(?:e)?|Jul(?:y)?|Aug(?:ust)?|Sep(?:tember)?|Oct(?:ober)?
|(Nov|Dec)(?:ember)?)\\s+\\d{4}”)}
```

G. FIND MISC FUNCTION

```
# This function finds all cases of text between the word “in” and
# the next punctuation. However, dashes are removed first so the
# regex does not terminate at a dash. It can accomodate more than
# one instance of this occurence within an event
# If the string found starts with the word “the,” “the” will be
# removed.

find.misc <- function(test){
test <- rm_dashes(paste0 (test, .”“))
if (sum(grepl(“\\sin\\s,”test)) > 0){
misc <- gsub (“(.*?)\\<in\\> (.*)[[:punct:]](.*)?,” “\\2,”
test, ignore=T)
gimme <- regexpr (“\\<in\\>,” test, ignore=T)

if (sum(gimme>0) > 0){
test <- sub (“\\<in\\>,” “XX,” test, ignore=T)
gimme <- regexpr (“\\<in\\>,” test, ignore=T)
misc <- append(misc,gsub (“(.*?)\\<in\\> (.*)[[:punct:]]
(.*)?,” “\\2,” test[gimme > 0], ignore=T))
}
test <- sub (“\\<in\\>,” “XX,” test, ignore=T)
gimme <- regexpr (“\\<in\\>,” test, ignore=T) # reset
misc <- append(misc,gsub (“(.*?)\\<in\\> (.*)[[:punct:]]
(.*)?,” “\\2,” test[gimme > 0], ignore=T))
paste(gsub(‘.*the ‘, ‘‘,misc),collapse = ,” “)
}
else{
NA
}
}
```

H. HAMMING LOCATION SEARCH R SCRIPT

```
#!/usr/bin/Rscript

#PBS -l procs=300
#PBS -l pmem=1GB
#PBS -l walltime=23:00:00

load(“hammingdfs.RData”)

require(parallel)
find.loc  <-  function  (i,  places  =  unlisted_df[,1],  text  =
df3$text2) {
patt <- paste0 (“\\b,” places[i], “\\b”)
hit <- grep (patt, text,ignore.case = TRUE)
if (length (hit) == 0)
return (c(0, 0))
return  (c(hit[1],  length(hit)))  #returns  first  hit  and  total
number of hits
}
cat(date())
ptm = proc.time()
# Try with entire unlisted_df alternate names column
search_all  <-  do.call(rbind,  mclapply(seq(1:nrow(unlisted_df)),
function(x) find.loc(x)))

time_len = proc.time() - ptm
cat(time_len)
cat(date())

save(search_all,time_len, file=“search_results.RData”)
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Bates D, Maechler M (2015) Matrix: Sparse and dense matrix classes and methods. R package version 1.2-3. http://CRAN.R-project.org/package=Matrix.

Bunzel C (2015) From paper state to caliphate: The ideology of the Islamic State. *Analysis Paper,* 19:32-34.

Burnham G, Lafta R, Doocy S, Roberts L (2006) Mortality after the 2003 invasion of Iraq: A cross-sectional cluster sample survey. *The Lancet*, 368(9545): 1421–1428.

Chang W, Cheng J, Allaire JJ, Xie Y, McPherson J (2016) Shiny: Web application framework for R. R package version 0.13.2. https://CRAN.R-project.org/package=shiny.

Cheng J, Xie Y (2016) Leaflet: Create interactive web maps with the JavaScript “Leaflet” Library. R package version 1.0.1. https://CRAN.R-project.org/package=leaflet.

Combating Terrorism Center (CTC) (2006) Harmony Program. Accessed January 5, 2016, https://www.ctc.usma.edu/programs-resources/harmony-program.

Faraway JJ (2006) *Extending the Linear Model with R Generalized Linear, Mixed Effects and Nonparametric Regression Models* (Taylor & Francis Group, Boca Raton, FL).

Feinerer I, Hornik K (2015) Text mining package “tm,” Version 0.6-2. (Jul 3) https://cran.r-project.org/web/packages/tm/tm.pdf.

Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 1–22.

GeoNames Geographical Database. Iraq(IQ) locations. Accessed May 1, 2016, http://download.geonames.org/export/dump/.

Glyph & Cog, LLC. (2014) Xpdf, Open source viewer for PDF file. Accessed January 17, 2016, http://www.foolabs.com/xpdf/.

Global Terrorism Database (GTD) Overview of the GTD. Accessed January 17, 2016, http://www.start-dev.umd.edu/gtd/.

Hanin Network Forums (2014) Iraq: ISIL claims 302 attacks on Iraqi forces in Al-Anbar from 2 to 30 January. Retrieved on February 3, 2016, https://www.opensource.gov.

Hastie T, Tibshirani R, Friedman J (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. (Springer New York Inc., New York).

Hornik K, Buchta C, Zeileis A (2009) Open-source machine learning: R meets weka. *Computational Statistics*, 24(2), 225–232.

Iraq Body Count (IBC) How Accurate Is IBC? Accessed April 5, 2016, http://www.iraqbodycount.org/analysis/beyond/state-of-knowledge/7.

Jensen, M (2013) Discussion Point: The Benefits and Drawbacks of Methodological Advancements in Data Collection and Coding: Insights from the Global Terrorism Database (GTD). Retrieved on February 5, 2016, http://www.start-dev.umd.edu/start/announcements/announcement.asp?id=622.

Kaplan F (2006) Number Crunching. Taking another look at the Lancet's Iraq study *Slate* (October 20), http://www.slate.com/articles/news_and_politics/war_stories/ 2006/10/number_crunching.html

Kumar S (2014) Docx to Text convertor. Retrieved from http://docx2txt.sourceforge.net/.

LaFree G, Dugan L (2007) Introducing the Global Terrorism Database, Terrorism and Political Violence, 19:2, 181–204.

Laub Z, Masters J (2013) "The Islamic State" Council on Foreign Relations. Accessed March 18, 2016, http://www.cfr.org/iraq/islamic-state/p14811.

Moore SE (2006) 655,000 War Dead? A bogus study on Iraq casualties. *Wall Street Journal* (October 19), http://www.opinionjournal.com/editorial/feature.html?id= 110009108.

Pape R, Ruby K, Bauer V, Jenkins G (2014) How to fix the flaws in the Global Terrorism Database and why it matters. *The Washington Post* (Aug 11), https://www.washingtonpost.com/news/monkey-cage/wp/2014/08/11/how-to-fix-the-flaws-in-the-global-terrorism-database-and-why-it-matters/.

Porter, M (1980) An algorithm for suffix stripping, *Program*, 14(3):130-137.

R Core Team (2015) R: A language and environment for statistical computing [Computer software]. Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org.

Sloboda J (2006) On Iraq Body Count: A presentation by IBC cofounder John Sloboda at a working group meeting on methodologies used by researchers to estimate numbers of armed conflict deaths (Feb 17).

SMART information retrieval system (2004) English stopwords (Aug 31) http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop.

Spagat, M (2010) Ethical and data-integrity problems in the second lancet survey of mortality in iraq. *Defence and Peace Economics*. 21(1): 1–41.

Van Os A (2008) Antiword: A free MS Word document reader. Retrieved March 18, http://www.winfield.demon.nl/.

Viola P, Jones MJ (2004) Robust real-time face detection. *International Journal of Computer Vision.* 57(2): 137–154.

Weiss SM, Indurkhya N, Zhang T (2010) *Fundamentals of Predictive Text Mining* (Springer New York Inc., New York).

Whiteside C (2014) The smiling scented men: The political worldview of the Islamic State of Iraq, 2003–2013 (December) Doctoral dissertation, Washington State University.

Whiteside C (2016) Islamic State of Iraq dissertation database information provided via personal communication with author on, February 16. Monterey, California.

Wickham H (2015). stringr: simple, consistent wrappers for common string operations. R package version 1.0.0. https://CRAN.R-project.org/package=stringr.

Winter C (2015) *Documenting the Virtual "Caliphate"* (Quilliam, London).

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California